



UNIVERSIDADE DA BEIRA INTERIOR  
DEPARTAMENTO DE INFORMÁTICA

SEBENTA

---

**Multimédia**

---

*Autor:*  
Manuela PEREIRA

Última actualizaçāo: 20 de Setembro de 2022



# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>1</b>
1.1	Multimédia . . . . .	1
1.1.1	Digitalização . . . . .	2
1.1.2	Compressão . . . . .	2
1.1.3	Sincronização . . . . .	2
1.1.4	Interacção . . . . .	2
1.2	Resumindo . . . . .	2
<b>2</b>	<b>Digitalizar</b>	<b>5</b>
2.1	Representação digital dos dados . . . . .	5
2.1.1	Som . . . . .	6
2.1.2	Imagen . . . . .	7
2.2	Processo de digitalização . . . . .	8
2.2.1	Amostragem . . . . .	9
2.2.2	Quantificação . . . . .	10
2.2.3	Codificação . . . . .	14
2.3	Conversão analógico - digital . . . . .	15
2.4	Conversão digital - analógico . . . . .	15
2.5	Digitalização de texto . . . . .	16
2.6	Conclusões . . . . .	16
2.7	Problemas . . . . .	17
<b>3</b>	<b>Representação de Áudio Digital</b>	<b>19</b>
3.1	Pulse Code Modulation (PCM) . . . . .	19
3.2	Algoritmos de compressão de som . . . . .	20
3.2.1	Modulações diferenciais . . . . .	20
3.2.2	Algoritmos generalistas não destrutivos . . . . .	21
3.2.3	Algoritmos destrutivos: a codificação Psico-acústica . . . . .	21
3.2.4	MP3 . . . . .	23
3.3	Formatos de ficheiros de áudio digital . . . . .	24
<b>4</b>	<b>Cor e Codificação da cor</b>	<b>27</b>
4.1	A luz e a cor . . . . .	27
4.2	A percepção das cores . . . . .	28
4.3	Os modelos de cor . . . . .	29
4.3.1	Modelo RGB . . . . .	30
4.3.2	modelo CMY . . . . .	32
4.3.3	modelo CMYK . . . . .	33

4.3.4	Modelo HSV (Hue, Saturation, Value) . . . . .	33
4.3.5	Modelos CIE . . . . .	34
4.3.6	Modelos que separam luminância de crominância . . . . .	39
4.4	Gamma e modo R'G'B' . . . . .	40
4.5	Paletes de cor e cor indexada . . . . .	41
4.6	Conclusões . . . . .	45
4.7	Problemas . . . . .	45
<b>5</b>	<b>Imagen Digital</b>	<b>47</b>
5.1	Imagen em mapa de pontos . . . . .	47
5.1.1	Tamanho das imagens . . . . .	47
5.1.2	Resolução da imagem . . . . .	48
5.1.3	Qualidades e defeitos das imagens em mapa de pontos . . . . .	49
5.1.4	Formatos de imagem em mapa de pontos . . . . .	49
5.2	Imagen vectorial . . . . .	54
5.2.1	Qualidades e defeitos das imagens vectoriais . . . . .	54
5.2.2	Formatos da imagem vectorial . . . . .	55
5.2.3	SVG: o HTML do vectorial . . . . .	55
5.2.4	Meta-formatos . . . . .	56
5.3	Problemas . . . . .	57
<b>6</b>	<b>Codificação Fonte</b>	<b>59</b>
6.1	Redundância e quantificação da redundância . . . . .	60
6.1.1	Incerteza vs. informação . . . . .	60
6.1.2	Entropia . . . . .	61
6.2	Os parâmetros da compressão . . . . .	62
6.3	Técnicas de base . . . . .	63
6.3.1	Run Length Coding (RLC) . . . . .	63
6.3.2	Codificação relativa . . . . .	64
6.4	Algoritmos de codificação estatística . . . . .	64
6.4.1	Método de Huffman . . . . .	64
6.4.2	Método de Huffman canónico . . . . .	72
6.4.3	Método de Huffman adaptativo . . . . .	73
6.4.4	Método de Shannon-Fano . . . . .	79
6.4.5	Codificação Aritmética . . . . .	80
6.5	Algoritmos do tipo dicionário . . . . .	83
6.5.1	Algoritmo de codificação LZW . . . . .	83
6.5.2	Algoritmo de descodificação LZW . . . . .	88
6.6	Os métodos mistos: estatísticos e de dicionário . . . . .	91
6.7	Conclusões . . . . .	91
6.8	Problemas . . . . .	92
<b>7</b>	<b>Representação de Imagem Digital</b>	<b>97</b>
7.1	Medidas de distorção . . . . .	97
7.2	Sistema de compressão . . . . .	98
7.2.1	Transformada de cor . . . . .	99
7.2.2	Transformadas . . . . .	99
7.2.3	Frequência espacial e a DCT . . . . .	103
7.2.4	Definição de DCT . . . . .	103
7.2.5	Definição de 2D DCT . . . . .	104

7.2.6	Propriedades da DCT . . . . .	106
7.2.7	DCT versus DFT/KLT . . . . .	110
7.2.8	DCT e quantização . . . . .	110
7.2.9	Wavelets . . . . .	110
7.2.10	Quantização . . . . .	115
7.2.11	Quantização escalar vs. vectorial . . . . .	116
7.2.12	Quantização escalar . . . . .	116
7.2.13	Quantização não uniforme . . . . .	118
7.3	A norma JPEG . . . . .	119
7.3.1	Modos de operação . . . . .	119
7.3.2	Principais passos da compressão JPEG . . . . .	120
7.3.3	DCT em blocos . . . . .	121
7.3.4	Quantização . . . . .	121
7.3.5	Funcionamento da DCT e da Quantização . . . . .	122
7.3.6	Codificação entrópica . . . . .	128
7.3.7	JPEG: Modo de codificação progressivo . . . . .	130
7.3.8	JPEG: Modo de codificação sem perdas . . . . .	131
7.3.9	JPEG: Modo de codificação hierárquico . . . . .	132
7.3.10	Relação Débito-Qualidade . . . . .	132
7.3.11	MJPEG (Motion JPEG) . . . . .	133
<b>8</b>	<b>Representação de Vídeo Digital</b>	<b>135</b>
8.1	Vídeo analógico . . . . .	135
8.2	Vídeo digital . . . . .	137
8.2.1	Standard (ITU-R-601) . . . . .	137
8.2.2	Outros standards . . . . .	138
8.3	Compressão de vídeo . . . . .	139
8.3.1	Compressão temporal . . . . .	140
8.3.2	Norma MPEG . . . . .	141
8.3.3	MPEG-1 . . . . .	142
8.3.4	MPEG-2 . . . . .	146
8.3.5	MPEG-4 . . . . .	147
8.3.6	Codificação das texturas dos VOP . . . . .	153
8.3.7	H.264 - Codificação avançada de vídeo . . . . .	154
8.3.8	Os formatos de ficheiro de vídeo digital . . . . .	155
8.3.9	MPEG-7 . . . . .	157
8.3.10	MPEG-21 . . . . .	158



# **Lista de Figuras**

2.1	Sinais acústicos com diferentes frequências e a mesma amplitude.	6
2.2	Sinais acústicos com a mesma frequência e diferentes amplitudes.	6
2.3	Sinais acústicos com frequências de 4HZ (esquerda) e 14Hz (direita).	7
2.4	Sinais acústicos. O sinal representado a azul tem duas vezes a amplitude do sinal representado a vermelho. . . . .	7
2.5	Representação de uma imagem bitmap. . . . .	8
2.6	Representação de diferentes intensidade por diferentes valores numéricos. . . . .	8
2.7	Definição das amostras de um sinal acústico. . . . .	9
2.8	Representação de uma imagem como uma matriz de pontos. . . .	10
2.9	Representação de uma imagem num gráfico de intensidade. . . .	10
2.10	Representação de uma imagem em diferentes resoluções. . . . .	11
2.11	Representação de uma imagem em diferentes rresoluções. . . .	11
2.12	Representação de uma imagem em diferentes profundidades de cor.	12
2.13	Representação de uma imagem com 2, 4, 8, 16, 32, 64, 128 e 256 níveis de cinzento. . . . .	13
2.14	Definição dos intervalos de quantificação de um sinal de áudio. .	14
2.15	Sinal a digitalizar. . . . .	15
2.16	Conversão analógico - digital. . . . .	15
2.17	Conversão digital - analógico. . . . .	16
3.1	Sinal original (frequência = 1Hz); Frequência de amostragem = 1Hz; Frequência de amostragem = 1.5 Hz. . . . .	19
3.2	Curvas de resposta do ouvido humano. . . . .	22
4.1	Espectro da luz. . . . .	27
4.2	Luz absorvida. . . . .	28
4.3	Cores primárias e secundárias no modelo de cor RGB. . . . .	31
4.4	Representação do modelo de cor RGB. . . . .	31
4.5	Cores primárias e secundárias no modelo de cor CMY. . . . .	32
4.6	Representação do modelo de cor CMY. . . . .	33
4.7	Representação do espaço HSV. A: Hue; B: Value; C: Saturation..	34
4.8	Representação do modelo HSV. Na esquerda com variações no Hue. Na direita com a saturação e o valor a variarem para um tom específico. . . . .	34
4.9	Variação da saturação e do valor para um tom vermelho específico.	35
4.10	Modelo de cor CIE. . . . .	35
4.11	Modelo de cor CIE mapeado nas coordenadas X e Y. . . . .	36

4.12	Esquerda: <i>Color-matching functions</i> observadas no standard da CIE. Direita: <i>Color matching functions</i> do CIE 1931 RGB. Estas funções são a quantidade de primárias necessárias para obter a cor teste sobre as primárias. . . . .	36
4.13	Esquerda: representação do modelo RGB sobre a representação do modelo XYZ. Direita: Representação das gamas de cor representáveis por um modelo de cor RGB e por um modelo de cor CMY. . . . .	37
4.14	xyY. . . . .	37
4.15	Atlas de Albert H. Munsell. . . . .	38
4.16	Representação do modelo CIE LAB. . . . .	39
4.17	Sem correcção Gamma (esquerda); Com correcção Gamma (direita). . . . .	40
4.18	Cores representadas sem e com a correcção gamma.. . . . .	41
4.19	Imagen representada usando uma paleta exacta. . . . .	42
4.20	Imagen em cor indexada. . . . .	42
4.21	Da esquerda para a direita: palete do Windows; palete do Macintosh; palete do WEB; 3 Canais RGB. . . . .	43
4.22	Representação de uma imagem em usando diferentes tipos de paletes. . . . .	43
4.23	Representação de uma imagem em usando diferentes tipos de paletes. . . . .	44
5.1	Visualização da imagem baboon em formato gif entrelaçado. . . . .	51
5.2	Visualização da imagem baboon em formato PNG entrelaçado. . . . .	53
5.3	Imagen vectorial. . . . .	56
7.1	Esquema de compressão descompressão. . . . .	97
7.2	Sistema de compressão. . . . .	99
7.3	YCbCr em 4:4:4, em 4:2:2 em 4:1:1 e em 4:2:0. . . . .	100
7.4	Imagens originais. . . . .	100
7.5	Histogramas. . . . .	101
7.6	Auto-correlação normalizada entre os pixeis. . . . .	101
7.7	Funções de base da DCT. . . . .	105
7.8	2D DCT. . . . .	106
7.9	(a) Auto-correlação normalizada de uma imagem não correlacionada, antes e depois da DCT. (b) Auto-correlação normalizada de uma imagem correlacionada, antes e depois da DCT. . . . .	107
7.10	(a) Imagem não correlacionada e a sua DCT. (b) Imagem correlacionada e a sua DCT. . . . .	108
7.11	(a) Imagem "Saturn"e a sua DCT; (b) Imagem "Child"e a sua DCT; (c) Imagem "Circuit"e a sua DCT. . . . .	108
7.12	(d) Imagem "Trees"e a sua DCT; (e) Imagem "Baboon"e a sua DCT; (f) Imagem da onda seno e a sua DCT. . . . .	109
7.13	DCT inversa da imagem "Saturn"; (a) 100%; (b) 75%; (c) 50%; (d) 25%. . . . .	111
7.14	DCT inversa da imagem "Child"; (a) 100%; (b) 75%; (c) 50%; (d) 25%. . . . .	111
7.15	DCT inversa da imagem "Circuit"; (a) 100%; (b) 75%; (c) 50%; (d) 25%. . . . .	112

7.16	DCT inversa da imagem "Trees"; (a) 100%; (b) 75%; (c) 50%; (d) 25%. . . . .	112
7.17	DCT inversa da imagem "Baboon"; (a) 100%; (b) 75%; (c) 50%; (d) 25%. . . . .	113
7.18	DCT inversa da onda seno; (a) DCT(100%); (b) DCT(75%); (c) DCT(50%); (d) DCT(25%). . . . .	113
7.19	Flor Original e com 16, 8 e 4 coeficientes de DCT. . . . .	114
7.20	Decomposição piramidal de uma imagem. . . . .	114
7.21	Decomposição da imagem "Lena"em três níveis. . . . .	115
7.22	Quantização escalar. . . . .	116
7.23	Quantização vectorial. . . . .	116
7.24	Duas representações diferentes do mesmo quantizador escalar. . . . .	117
7.25	O caso mais geral de quantização escalar. . . . .	117
7.26	Quantização escalar "Midrise"vs "Midtread". . . . .	118
7.27	Esquema da codificação JPEG. . . . .	120
7.28	Imagen Lena com a representação de dois blocos de $8 \times 8$ . . . . .	122
7.29	Imagen Lena com diferentes níveis de quantização. . . . .	128
7.30	Processo de linearização segundo uma sequência em zigzag. . . . .	128
7.31	Esquema do modo de codificação sem perdas do JPEG. . . . .	131
7.32	Esquema do codificador JPEG hierárquico. . . . .	133
7.33	Imagen Lena comprimida a diferentes débitos. Cima, esquerda: 65536 Bytes (8 bpp); Cima, direita: 4839 Bytes (0.59 bpp); Baixo, esquerda: 3037 Bytes (0.37 bpp); Baixo, direita: 1818 Bytes (0.22 bpp). . . . .	133
7.34	Imagen com cor, original e depois de passar pelo processo de compressão descompressão JPEG. . . . .	134
8.1	Calculo de uma frame B. . . . .	144
8.2	Organização das frames no MPEG. . . . .	145
8.3	Segmentação de uma cena em objectos. . . . .	148
8.4	Decomposição dos objectos. . . . .	149
8.5	Descrição da cena. . . . .	150
8.6	Codificação baseada nos objectos vs. codificação baseada em blocos. . . . .	151
8.7	Codificação baseada nos objectos. . . . .	152
8.8	Técnica de "sprites". . . . .	152
8.9	Objecto sintetizados. . . . .	153



# Capítulo 1

## Introdução

### 1.1 Multimédia

Fazendo a análise etimológica do conceito Multimédia temos: *multus* (muitos) + *medium* (meio). Neste sentido multimédia significa a utilização de vários meios de difusão de informação. Quando este termo é usado como nome, por exemplo "O multimédia..." significa que são usados vários meios para um mesmo produto de comunicação ou de entretenimento. Neste caso um produto multimédia é uma associação de vários meios sobre um mesmo suporte (por exemplo um CDROM ou um DVD): Texto; Imagem fixa; Imagem animada; Som; Vídeo.

Esta noção é no entanto demasiado simplista não fazendo qualquer referência à forma como são criados, difundidos ou consultados os conteúdos multimédia. Segundo estas definições teríamos de considerar multimédia um jornal, que apresenta informação por intermédio de vários meios, tais como o texto e as ilustrações ou uma televisão, que mistura som e texto com imagens em movimento.

Quando falamos de multimédia no âmbito desta disciplina estamos interessados em meios digitais, relacionados com a manipulação de informação digitalizada e controlada por computador. Os sistemas e aplicações multimédia combinam os seguintes tipos de informação multimédia:

- Texto.
- Imagem fixa.
- Imagem animada.
- Som.
- Vídeo.

Nos sistemas multimédia, todos estes media são representados sob a forma digital, isto é, são codificados por meio de dígitos binários ou bits. Contudo, eles distinguem-se entre si pois a interpretação que é dada aos dígitos binários varia conforme a natureza da sua apresentação.

Quando falamos em meios como a imagem, o áudio ou o vídeo em formato digital teremos obrigatoriamente de falar de compressão. Meios como o áudio e o vídeo são dependentes do tempo, obrigando a falar de sincronização temporal.

Sendo estes dados digitais podem ser difundidos sobre redes de comunicação ou guardados sobre suportes numéricos, podemos também pensar em diversos tipos de interacção que passam a ser possíveis ao utilizador de um produto multimédia.

### 1.1.1 Digitalização

Todas as informações colocadas em conjunto numa aplicação multimédia estão obrigatoriamente sob a forma digital. O que significa que são traduzidas em linguagem binária. O computador é a ferramenta de produção, tratamento, armazenamento, comunicação e consulta.

Com estas noções em mente facilmente compreendemos que:

- realizar um produto multimédia, é realmente criar, tratar, guardar, estruturar, ligar e sincronizar ficheiros binários.
- difundir um produto multimédia sobre um suporte do tipo CDROM ou sobre uma rede de comunicação é sempre distribuir ficheiros binários.
- ler um ficheiro multimédia é descodificar ficheiros binários para permitir a representação analógica sobre um ecrã (texto, imagem fixa ou animada, vídeo) ou a restituição analógica para um altifalante (som).

### 1.1.2 Compressão

Toda a informação colocada numa aplicação multimédia é não só digitalizada mas também comprimida. (25fps, 640x480 pixéis, 24 bpp: 1 segundo = 22 Mo, 1 min = 2 CDROM).

### 1.1.3 Sincronização

Existe uma diferença essencial entre os diferentes meios (texto, imagem fixa, imagem animada, vídeo, som) que implica um tratamento e sincronização específicos. Texto e imagem fixa são independentes do tempo enquanto que imagem animada, vídeo e som são dependentes do tempo. Estes últimos obrigam a uma sincronização temporal.

### 1.1.4 Interacção

O acesso às informações existentes numa aplicações multimédia implica a criação de procedimentos de interacção entre o leitor e o produto.

## 1.2 Resumindo

Um produto multimédia pode ser definido como uma combinação de meios digitais:

- discretos e contínuos;
- sincronizados e ligados de forma a permitir ao utilizador agir sobre o desenrolar da aplicação e de escolher o seu itinerários de consulta (interactividade);

- que podem ser difundidos sobre redes de telecomunicações ou sobre qualquer suporte numérico (CDROM, DVD, etc.....).



# Capítulo 2

## Digitalizar

Criar um produto multimédia é tratar, estruturar e sincronizar, sobre um mesmo suporte textos, sons, imagens fixas e animadas, permitindo ao futuro utilizador possibilidades reais de interacção, particularmente em termos de itinerário e ritmo de consulta.

Juntar e sincronizar sobre um mesmo suporte médias assim tão diferentes implica necessariamente que eles partilhem o mesmo modo de representação. Assim eles poderão ser interpretados por um mesmo suporte: computador, PDA, telemóvel, etc.

### 2.1 Representação digital dos dados

Digitalizar uma informação significa representar essa informação usando uma representação binária, isto é, apenas 0s e 1s.

**Nota** Para recordar:

1 byte	8 bits
1 Kb	1024 bits
1 Mb	1024 Kb
1 Gb	1024 Mb
1 Tb	1024 Gb
1 KB	1024 bytes
1 MB	1024 KB
1 GB	1024 MB
1 TB	1024 GB

Tabela 2.1: Conversões binárias

Acabamos de ver que em multimédia teremos o som, o vídeo, a imagem, etc. digitalizados, isto é, em formato digital. Duas perguntas lógicas são: Porquê? e Como?

- Porquê? Computador.

- Como? Transformando uma variação contínua (de sons, de níveis de cinzento, de cores, etc.) numa sucessão de elementos descontínuos cujas características quantificáveis se possam representar em formato digital.

Assim teremos de responder a mais uma questão:

Quais são as características quantificáveis de um som ou de uma imagem?

### 2.1.1 Som

Vamos começar por definir um som para podermos compreender como o podemos digitalizar. Vimos que para isso teremos de encontrar as características quantificáveis do som.

Sendo um som, ou sinal acústico, uma variação da pressão do ar ao longo do tempo, podemos defini-lo usando dois valores:

- A frequência das ondas sonoras emitidas por um instrumento de música define fisicamente a altura e o timbre do som (ver figura 2.1);



Figura 2.1: Sinais acústicos com diferentes frequências e a mesma amplitude.

- A amplitude dessas mesmas ondas define a potência (ver figura 2.2);

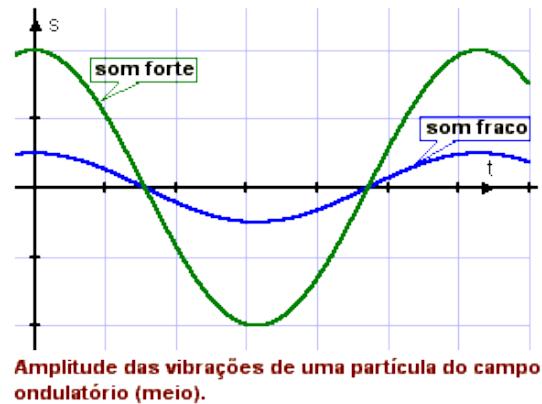


Figura 2.2: Sinais acústicos com a mesma frequência e diferentes amplitudes.

Ambos estes dados são dados puramente quantitativos. Eles designam respectivamente:

**Frequência** número de vibrações por unidade de tempo. A duração de uma vibração define o seu período ( $T$ ). Por exemplo 1Hertz (Hz) = 1 vibração por segundo; 100 Hz = 100 vibrações por segundo. O período é respectivamente de 1 segundo e de  $\frac{1}{100}$  de segundo. A largura da onda ( $\lambda$ ) designa o espaço percorrido por uma onda ao longo de um período ( $\lambda = V \times T$ , velocidade de propagação x tempo periódico). A largura da onda é inversamente proporcional à frequência ( $f = 1/T$ ).

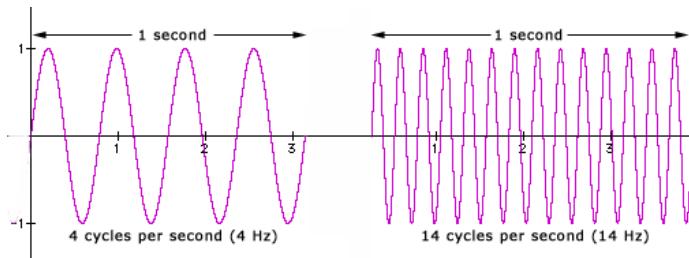


Figura 2.3: Sinais acústicos com frequências de 4Hz (esquerda) e 14Hz (direita).

**Amplitude** valor máximo do desvio em relação ao ponto central do movimento vibratório.

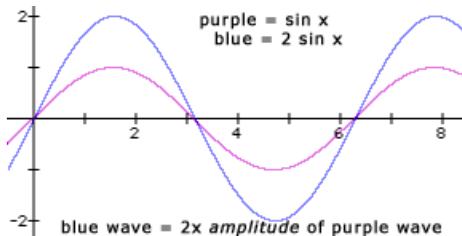


Figura 2.4: Sinais acústicos. O sinal representado a azul tem duas vezes a amplitude do sinal representado a vermelho.

### 2.1.2 Imagem

Uma imagem pode ser considerada como um conjunto de pontos coloridos, ou em níveis de cinzento. Estas imagens concebidas como um conjunto de pontos são chamadas de bitmaps.

Qualquer cor pode ser definida pela quantidade de cada uma das cores primárias que ela contém. Ou no caso de imagens ditas a preto e branco, é fácil imaginar uma escala quantitativa que vai do preto ao branco passando por um certo número de degradados intermédios.

Também podemos considerar que uma imagem é feita não de pontos, mas de objectos predefinidos, como quadrados, círculos, elipses, segmentos de recta, etc. Estes objectos podem receber designações numéricas de acordo com características quantitativas tais como tamanho, posição, cor, etc. Dizemos, neste caso, que a imagem é vectorial, o que significa que ela não é constituída por uma nuvem de pontos, mas por objectos definidos pelas suas numerosas características.

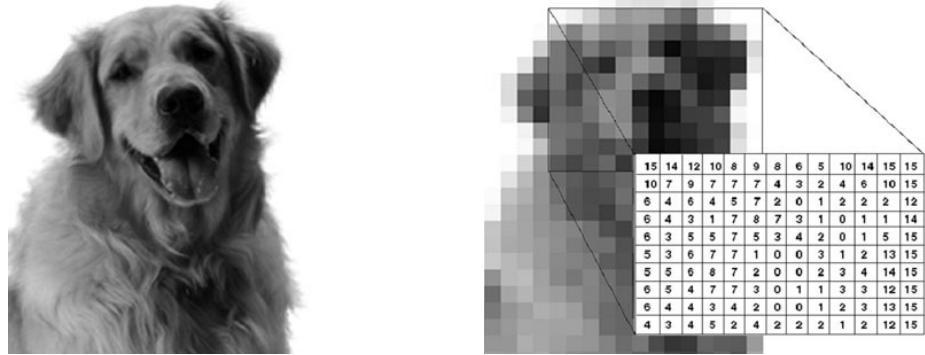


Figura 2.5: Representação de uma imagem bitmap.

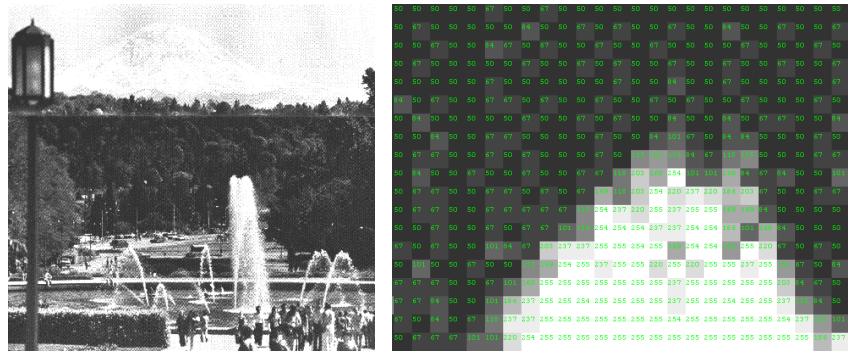


Figura 2.6: Representação de diferentes intensidades por diferentes valores numéricos.

Assim concluímos que tanto o som como as imagens podem ser representados por dados quantitativos.

Vamos ver de seguida qual o processo por que passa cada um destes media para que passe para uma representação digital. É o chamado processo de digitalização.

## 2.2 Processo de digitalização

A digitalização é caracterizada por três etapas:

- Amostragem;
  - Quantificação;
  - Codificação.

Vamos ver como se processa cada uma destas etapas.

### 2.2.1 Amostragem

Falamos no início deste capítulo que o princípio geral da digitalização consistia fundamentalmente na transformação de uma variação contínua numa sucessão de elementos descontínuos ou discretos, aos quais serão afectados valores numéricos. Esta passagem do contínuo ao discreto define a etapa da amostragem.

Como é feita esta fase no caso específico do som e da imagem é o que vamos ver de seguida.

#### Som

No caso do som não vamos partir o som em si mas sim a sua imagem eléctrica. É a imagem eléctrica da onda sonora que vai ser partida em amostras segundo o eixo do tempo, definindo assim os intervalos aos quais vamos proceder à medição da amplitude do sinal.

**Nota** O número de amostras por segundo define a **frequência da amostragem**.

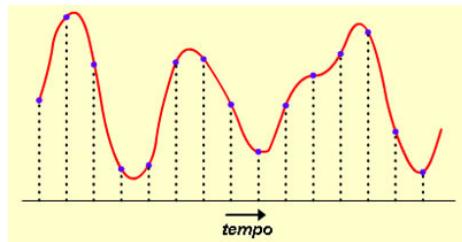


Figura 2.7: Definição das amostras de um sinal acústico.

Evidentemente, o número de amostras tomadas por segundo num sinal sonoro condiciona a qualidade de restituição final.

**Exemplo** Uma qualidade média de reprodução de sons por um telefone numérico necessita de uma amostragem de 8KHz (i.e. 8000 amostras por segundo).

Um registo numérico de qualidade hi-fi (por exemplo um CD de música) exige uma amostragem de 44.1 KHz (i.e. 44100 amostras por segundo).

#### Imagen

Uma imagem bitmap ou imagem em modo de pontos é constituída de um conjunto de pontos alinhados horizontal e verticalmente como linhas e colunas de uma tabela.

Um pixel é o elemento mais pequeno da imagem ao qual se pode associar uma cor. Todas as imagens reproduzidas num ecrã são imagens bitmap, da mesma forma são as imagens obtidas por um scanner ou por um aparelho fotográfico numérico.

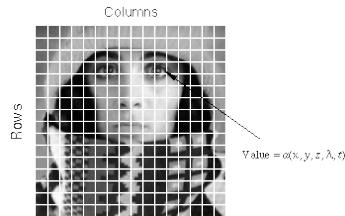


Figura 2.8: Representação de uma imagem como uma matriz de pontos.

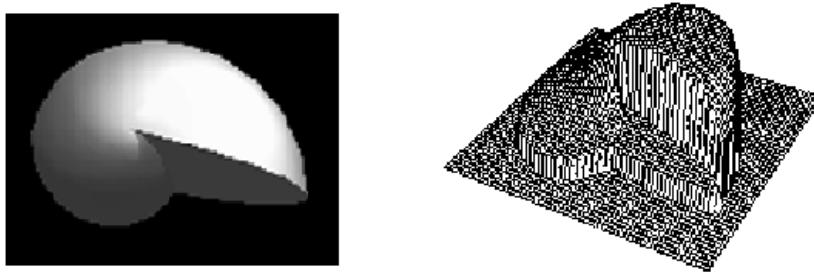


Figura 2.9: Representação de uma imagem num gráfico de intensidade.

**Nota** Evidentemente a resolução de uma imagem (número de pixels por unidade de largura) condiciona a qualidade final da imagem. Mas por outro lado um crescimento do número de pixels por unidade de largura aumenta a quantidade de informação a tratar e consequentemente o tamanho do ficheiro (mais espaço na memória e mais tempo de tratamento).

### 2.2.2 Quantificação

Estando as amostras definidas resta-nos atribuir-lhes um valor numérico, isto é quantificá-los, que é a mesma coisa que atribuir-lhes uma escala de medição.

#### Medida

O que vamos medir?

**Imagen** Nas imagens a informação quantitativa associada a cada pixel diz respeito à cor ou à densidade de cinzento no caso de imagens a preto e branco. Vamos por enquanto ficar pelas imagens a preto e branco. Para estas vamos ter de determinar, para os milhares de pixels que a constituem, a densidade específica de cada pixel. Mas o fluxo de luz varia de forma contínua, estes níveis de cinzento são potencialmente em número infinito, e desta forma susceptíveis de tomar uma quantidade infinita de valores. Ora um computador só pode tratar séries finitas de valores. Temos então de determinar por entre a quantidade teoricamente infinita de valores possíveis, os número de valores realmente possíveis de exprimir por um sistema informático.

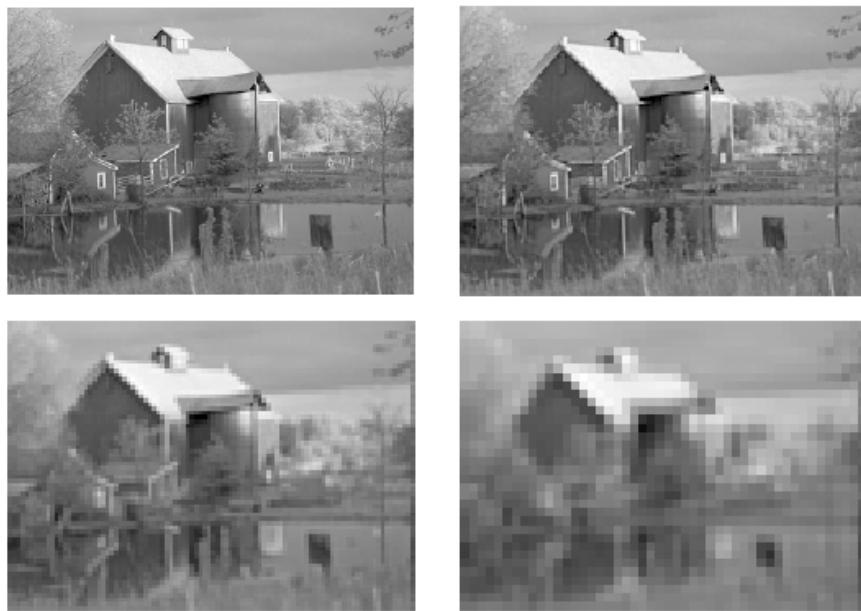


Figura 2.10: Representação de uma imagem em diferentes resoluções.



Figura 2.11: Representação de uma imagem em diferentes resoluções.

**Som** No caso do som, é a medida das amplitudes do sinal que vai fornecer os dados numéricos necessários. Várias milhares de vezes por segundo, vamos medir a altura do sinal referindo-a à escala correspondente sobre uma escala de medição.

### Escala de medição

Resta-nos então definir a escala de medição, isto é determinar o número de graduações que ela disporá. Isto trata-se noutrous termos de definir o seu passo de quantificação: quanto mais pequeno é o passo, menor é a distância entre duas

graduações e mais precisa é a medição.

Vemos imediatamente o interesse de uma escala de medida contendo o maior número possível de escalas. Mas o valor de cada escala deve ser designado por um número específico, utilizar um grande número de graduações implica que sejamos capazes de distinguir e consequentemente designar um grande número de valores.

O computador, como já vimos anteriormente, trabalha somente com binários. O problema é então determinar o número de bits que serão necessários para exprimir a quantidade de valores desejados.

**Exemplo** Quantos valores se poderão distinguir com apenas um bits?

Apenas dois, 0 ou 1. Por isso o valor de cada pixel de uma imagem constituída apenas de pontos pretos e brancos precisará apenas de um bit.

Com dois bits já conseguiremos distinguir quatro valores diferentes: 00, 01, 10, 11.

Com  $n$  bits distinguiremos  $2^n$  valores diferentes.

Assim sabemos que para distinguir  $N$  valores (para termos  $N$  graduações diferentes na nossa escala de medição) precisaremos de  $n = \log_2 N$  bits. Como  $n$  terá de ser um número inteiro, no caso de obtermos um número racional este terá de ser arredondado para o valor inteiro superior.

Por exemplo, para designar 256 graduações precisaremos de  $n = \log_2 256 = 8$  bits. Para 100 graduações precisaremos de  $n = \log_2 100 = 6.64$ , logo 7 bits.

As imagens representadas na figura 2.12 apresentam duas profundidades de quantificação diferentes. Na imagem da esquerda temos 256 graduações (logo 8 bits por pixel), enquanto na imagem da direita temos apenas 4 graduações (2 bits por pixel)



Figura 2.12: Representação de uma imagem em diferentes profundidades de cor.

Na figura 2.13 apresenta-se uma imagem com 8 profundidades de quantificação diferentes. Temos desde 2 graduações (apenas 1 bit por pixel, a imagem a preto e branco), 4 graduações (2 bits por pixel), 8 graduações (3 bits por pixel), etc. até 256 graduações (8 bits por pixel).



Figura 2.13: Representação de uma imagem com 2, 4, 8, 16, 32, 64, 128 e 256 níveis de cinzento.

Na figura 2.14 o sinal acústico está representado usando 8 valores diferentes (3 bits para representar cada amplitude).

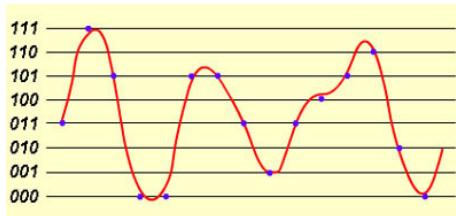


Figura 2.14: Definição dos intervalos de quantificação de um sinal de áudio.

**Exemplo** Um CD de música, por exemplo, recorre a uma codificação sobre 16 bits permitindo distinguir  $2^{16} = 65536$  níveis diferentes de amplitude.

**Nota** O número de bits utilizado para determinar as graduações de uma escala de medição condiciona a fineza da medida, e por conseguinte a qualidade de restituição final. Sempre que um valor da amostragem se situa entre duas graduações ele será aumentado ou diminuído de forma a corresponder ao valor da graduação mais próximo. Esta aproximação introduz o chamado erro de quantificação. Quanto maior é a profundidade da graduação mais fino é o passo de quantificação e menor será o erro de quantificação.

**Nota** A frequência de amostragem e a profundidade de quantificação são os dois factores que determinam a qualidade de um processo de digitalização

### 2.2.3 Codificação

A amostragem partiu o fenómeno a digitalizar em unidades elementares. A quantificação institui uma escala de medição que corta o conjunto de valores do sinal num certo número de graduações. A quantificação torna assim possível substituir cada uma das amostras pela indicação da sua correspondência numa dessas graduações.

Resta no entanto designar as diferentes graduações, isto é, dar a cada uma um nome que a distinga de todas as outras: esta operação é a operação de codificação.

Codificar consiste em estabelecer uma correspondência entre um conjunto de partida (aqui, o conjunto de pontos na escala de medição) e um conjunto de chegada composto de valores código (aqui, sequências binárias de tamanho uniforme).

A correspondência entre os elementos dos dois conjuntos deve satisfazer duas condições precisas:

1. Ele tem de ser uma função.

Relembrar: uma função impõe que cada elemento do conjunto de partida tem uma e uma só correspondência no conjunto de chegada.

2. Um código correcto exige que toda a palavra código seja a representação de no máximo um elemento do conjunto de partida.

Resumindo, a codificação é a definição de uma função injetiva de um conjunto a codificar num conjunto de palavras código. Codificar os valores quantificados significa associar-lhes uma sequência binária de tamanho uniforme. Já vimos anteriormente que determinamos esse tamanho como o número de bits necessário para a representação de todas as graduações da escala.

Terminamos assim a digitalização: o sinal analógico de partida tornou-se numa sequência binária que podemos manipular usando o computador.

## 2.3 Conversão analógico - digital

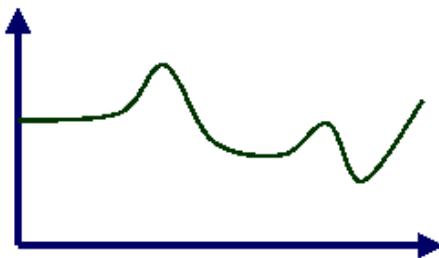


Figura 2.15: Sinal a digitalizar.

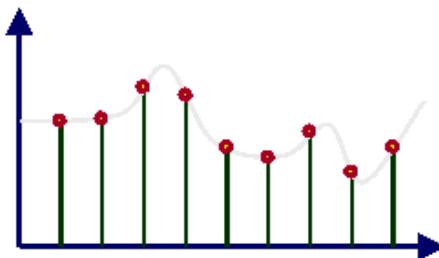


Figura 2.16: Conversão analógico - digital.

Relembremos alguns periféricos que terão obrigatoriamente de conter um conversor analógico-digital: carta de aquisição de vídeo, scanner, carta de som, rato, ecrã, leitor óptico (CDROM, DVD), leitor magnético (disco), e modem (na recepção).

## 2.4 Conversão digital - analógico

O sinal analógico quantificado, obtido na saída do conversor digital-analógico apresenta um aspecto de escada que tem de ser corrigido para se aproximar ao máximo do sinal analógico original. Para isso, é realizada uma interpolação, ou cálculo de valores intermédios para alisar o sinal.

Notemos que a digitalização de um sinal contínuo representa sempre uma degradação desse sinal.

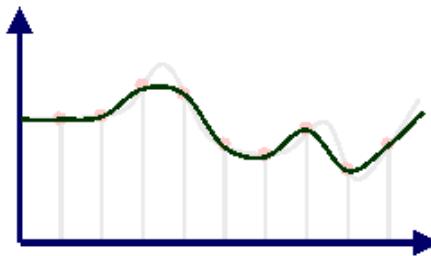


Figura 2.17: Conversão digital - analógico.

Relembremos alguns periféricos que terão obrigatoriamente de conter um conversor digital analógico: carta de som (na restituição), sintetizador de música, modem (na emissão).

## 2.5 Digitalização de texto

O texto foi o primeiro dos medias a ser digitalizado, mesmo antes da invenção da informática.

**1832** Código de Morse (comprimento variável).

**1874** Código de Baudot (comprimento constante de 5 bits, 2 caracteres especiais - permitindo assim  $32 \times 2$  símbolos).

No caso do texto a amostragem já esta realizada pelo alfabeto e a profundidade de quantificação imposta pela quantidade de símbolos a distinguir. Assim no caso do texto a digitalização limita-se a uma codificação.

Com a chegada da informática surgiu a necessidade de códigos mais ricos e funcionais que os códigos de Morse ou de Baudot.

**1963** Código ASCII (7 bits + 1 para controlo de erro): tem um total de  $2^7 = 128$  símbolos: maiúsculas, minúsculas, dígitos, vários símbolos especiais e caracteres de comando para controlo do terminal e para comunicação. Este não continha originalmente caracteres acentuados.

Código ASCII estendido: são utilizados os oito bits para determinar cada carácter. Já engloba os caracteres acentuados e um maior número de caracteres especiais e de controlo. É este que é utilizado actualmente nos nossos computadores.

**1991** Unicode (16 bits  $\rightarrow 2^{16} = 65536$  símbolos): código muito mais rico, integrando outros alfabetos além do latim (árabe, grego, hebraico, etc.).

## 2.6 Conclusões

Concluímos a digitalização notando que será sempre sobre dados numéricos que serão aplicados tratamentos matemáticos que nos permitirão modificar a

informação. É o chamado tratamento de texto, tratamento de som e tratamento de imagem.

Para terminar vejamos algumas vantagens e desvantagens entre um sinal analógico e um sinal digital.

- Analógico

- Sinal frágil.
  - Operações de tratamento delicadas. Estas são realizadas fisicamente sobre o sinal eléctrico com a ajuda de materiais pesados e caros.

- Digital

- Sinal estável.
  - Operações de tratamento puramente matemáticas.
  - Armazenamento seguro e fiável, graças aos progressos feitos no domínio das memórias.
  - Realização de redes multi-serviços: as técnicas adaptadas ao digital dizem respeito neste caso a qualquer dos medias: texto, som, imagem fixa, imagem animada, vídeo.

Terminamos assim a digitalização notando que foi a digitalização de todos os medias que permitiu o nascimento do multimédia.

## 2.7 Problemas

1. Uma fotografia de  $10 \times 20$  polegadas foi digitalizada tomando 300 amostras por polegadas (300 dpi) e usando 24 bits por amostra.  
Qual o espaço ocupado pela imagem digitalizada em KB.
2. Indique o que terá de ter em consideração quando escolhe o nível de amostragem.
3. Indique e explique os principais factores que determinam a qualidade de um processo de digitalização.
4. Explique porque é que a digitalização foi importante para o nascimento do multimédia.



## Capítulo 3

# Representação de Áudio Digital

### 3.1 Pulse Code Modulation (PCM)

Em áudio as fases de amostragem e quantização denominam-se PCM (Pulse Code Modulation). O teorema de Nyquist determina a frequência de amostragem exigida de forma a sermos capazes de reconstruir o som original. Para uma amostragem correcta devemos usar uma frequência de amostragem pelo menos igual a duas vezes a máxima frequência contida no sinal.

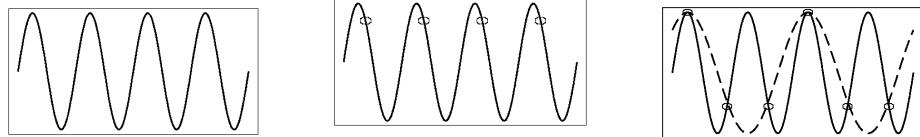


Figura 3.1: Sinal original (frequência = 1Hz); Frequência de amostragem = 1Hz; Frequência de amostragem = 1.5 Hz.

---

**Exemplo** Assumindo uma largura de banda para discurso de aproximadamente 50 Hz a 10 kHz, a taxa de Nyquist irá exigir uma amostragem de 20 kHz.

Usando quantização uniforme, o débito mínimo que conseguiremos será de aproximadamente 12 bits. Assim, para discurso em mono teríamos um débito de 240 kbps (30kB/sec).

Com quantização não uniforme, adaptando a quantização à sensibilidade do ouvido, podemos reduzir o débito para aproximadamente 8 bits por amostra guardando o nível de qualidade e reduzindo o débito para 160 kbps (20kB/sec).

O standard para a telefonia assume que a maior frequência que pretendemos reproduzir é de 4Khz. Desta forma a frequência de amostragem é de 8KHz e usando quantização não uniforme reduz este débito para 64 kbps (8kB/sec).

---

Quality	Sample Rate (KHz)	Bits por amostra	Mono/ Stereo	Data rate (KB/sec) (não comprimido)	Banda Frequência (KHz)
Telefone	8	8	Mono	8	0.200-3.4
Rádio AM	11.025	8	Mono	11.0	0.1-5.5
Rádio FM	22.05	16	Stereo	88.2	0.02-11
CD	44.1	16	Stereo	176.4	0.005-20
DAT	48	16	Stereo	192.0	0.005-20
DVD	192	24	6 canais	1.200.0	0-96
Audio	(max)	(max)		(max)	(max)

Tabela 3.1: Características de alguns tipos de áudio

## 3.2 Algoritmos de compressão de som

### 3.2.1 Modulações diferenciais

Este tipo de método propõe reduzir o número de bits necessários à quantização de cada amostra codificando apenas a diferença entre duas amostras sucessivas, em vez de codificar o valor de cada amostra.

#### Modulação delta

Trata-se aqui de substituir o valor que define uma amplitude de uma amostra por um único sinal (positivo ou negativo) representando a variação do sinal em relação ao seu valor precedente: esta informação necessita apenas de um único bit por amostra (1=diferença positiva, 0=diferença negativa).

Esta técnica, chamada modulação delta, supõe que cada valor quantificado só difere do valor precedente de mais ao menos 1 incremento de quantização.

Esta técnica é assim limitada aos sinais cujas variações são relativamente lentas. Variações rápidas excedem as capacidades de codificação do modulador e o sistema será assim incapaz de "seguir" o sinal de entrada com uma exactidão satisfatória: temos o que chamamos de saturação de declive.

#### DPCM (Differential Pulse Code Modulation)

Para tentar limitar a incidência da saturação de declive, vamos aqui codificar não apenas o sinal da variação, mas também vamos consagrar alguns bits supplementares à expressão da diferença entre o valor de uma amostra com o valor da mostra precedente.

Em geral a expressão desta diferença requer menos bits que a expressão do valor absoluto da amostra.: se por exemplo, considerarmos que numa série de amostras diferenças de +16 incrementos são muito pouco prováveis, uma codificação de 4 bits pode chegar para codificar as diferenças.

#### ADPCM (Adaptative Differential Pulse Code Modulation)

Neste caso vamos codificar a diferença entre duas amostras sucessivas (geralmente sobre 4 bits) e também vamos tomar em consideração a evolução do sinal sobre as amostras precedentes, para prever o valor óptimo do passo de quantização a utilizar, sempre com o objectivo de aumentar as capacidades de codificação

do modulador e evitar assim a saturação de declive, ou ao menos limitar a sua incidência.

Se, por exemplo, o sinal passa bruscamente de uma tensão elevada a uma tensão baixa, o valor do passo retido será grande; ao contrário, se o sinal de entrada apresenta variações de tensão baixas, o valor do passo retido será pequeno.

Teremos neste caso de guardar as variações dos passos de quantização numa tabela de passos. Este tipo e modulação diferencial é usado na telefonia digital sem fio doméstica (DECT: Digital Enhanced Cordless Telephone). Este método permite uma muito boa reprodução sonora com um débito inferior ao conseguido com um modulador PCM clássico.

### 3.2.2 Algoritmos generalistas não destrutivos

Relembremos que um algoritmo de compressão não destrutivo permite-nos encontrar, depois da descompressão, o ficheiro original na sua integridade: não há nenhuma perda de informação.

Relembremos também que o princípio geral deste tipo de compressão é fundado na procura de ocorrências múltiplas de uma mesma série de bytes, sendo estas séries guardadas num dicionário onde serão representadas por códigos, o mais curtos possível. Estes códigos substituirão de seguida no ficheiro comprimido as longas séries de bytes do ficheiro original. No entanto, se este tipo de compressão se mostra muito eficaz na compressão de texto (que integram sempre numerosas repetições), elas mostram-se pouco eficientes na compressão áudio visto que o discurso, a música ou os ruídos apresentam poucas sequências repetitivas.

Assim, os algoritmos generalistas do tipo WinZip, WinRaR, WinAce devem satisfazer-se de uma redução de volume do ficheiro - dependendo da complexidade da fonte- de 5% a 35%.

Os algoritmos não destrutivos especificamente dedicados ao som fazem bastante melhor: WavArc (.wa), RKAU (RK Audio, .rka) ou Perfect Clarity Audio (.pca) do célebre Sound Forge, permitem atingir ou mesmo passar, uma redução de volume de 50%.

No entanto estes resultados estão ainda longe de satisfazer as restrições de débitos impostas pela difusão em tempo real sobre a Internet, ou para as televisores ou rádios digitais sobre redes por cabo. Assim, temos necessidade de técnicas de compressão impondo desta vez uma inevitável perda de informação.

### 3.2.3 Algoritmos destrutivos: a codificação Psico-acústica

O princípio base da codificação Psico-acústica é simples: existe em todo o registo sonoro uma quantidade não negligenciável de sons que o ouvinte não percebe e ruídos que ele não quer ouvir. Suprimir as modulações "inúteis" e o ruído incomodo diminuirá o peso dos ficheiros sem, no entanto, degradar a qualidade de reprodução.

#### Propriedades do ouvido humano

A supressão de sons que o ouvido não pode perceber supõe uma análise fina da percepção auditiva humana.

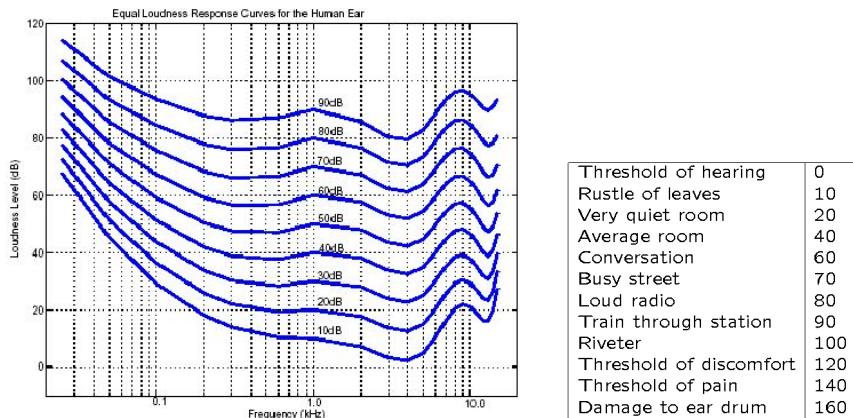


Figura 3.2: Curvas de resposta do ouvido humano.

- Em regra geral, suprimimos os sinais situados em frequências consideradas inúteis, e da mesma forma todos os sinais, qualquer que seja a sua frequência situados abaixo do limite de percepção.
  - Sabemos por exemplo, que a banda das frequências ditas audíveis, isto é 20Hz-20kHz, constitui um intervalo máximo ao qual a audição humana pode pretender em condições óptimas, e raramente atingidas. Assim, uma diminuição "razoável" da banda passante de um registo será a maior parte do tempo totalmente imperceptível.
  - Juntemos que neste intervalo de 20Hz - 20kHz, o intervalo absoluto de percepção não é uniforme: a sensibilidade do ouvido humano é máxima para frequências vizinhas de 3000 Hz e diminui consideravelmente de um lado e do outro de uma banda de 500Hz - 10 kHz. Assim, é inútil registrar os sons relativamente fracos com frequências inferiores a 100 Hz quando sabemos que o intervalo de audição nestas frequências é de 40 a 50 dB, quando que ele é inferior a 5dB à volta de 1000 a 5000 Hz.
  - Se juntarmos a esta constatação a pobre performance da maior parte dos sistemas de reprodução sonora de baixa e média gama quando se trata de reproduzir som no extremo grave, concluímos o quão inútil é aumentar os ficheiros destinados a uma difusão sobre Internet ou sobre os canais numéricos de rádio ou de televisão.
  - Pelas mesmas razões negligenciaremos as frequências superiores a 10 ou 11 KHz.
- Mas esta limitação absoluta da percepção auditiva em função das frequências não é a única de que sofre o sistema auditivo humano. Existe de facto outra limitação, esta relativa, que consiste na incapacidade de distinguir um som "coberto" por um som de frequência vizinha e de amplitude superior. É o famoso efeito de máscara.

Se, num grupo de frequências idênticas ou vizinhas, algumas têm uma amplitude muito superior às outras, apenas elas serão percebidas. Pode-

mos assim, sem grande sacrifício da qualidade de reprodução, suprimir as frequências mascaradas.

- Juntemos ainda a estes procedimentos, cuja exploração autoriza o essencial das economias realizadas na compressão destrutiva, uma terceira possibilidade de redução do tamanho de um ficheiro de áudio. Esta redução é também ela ligada às limitações da percepção auditiva, mas dizendo respeito desta vez à localização do som no espaço de difusão.

Sabemos que a estereofonia, graças a um registo e a uma difusão específicas dos canais direito e esquerdo, permitem ao ouvinte localizar a origem dos sons. É assim que um bom registo e um sistema de reprodução de alta qualidade permitem reconstruir virtualmente uma orquestra numa sala de audição. Não se trata realmente de uma capacidade do ouvido humano mas do cérebro humano, capaz de interpretar as muito pequenas diferenças que distinguem os sinais percebidos pelo ouvido esquerdo e pelo ouvido direito. Ora, esta capacidade é limitada às frequências superiores a algumas centenas de Hz. Vemos imediatamente a exploração que poderá ser feita desta particularidade. É de facto inútil, pelo menos nas frequências baixas, distinguir um canal de direita e um canal de esquerda. Vamos registrar apenas a parte grave do espectro em mono-fonia, realizando assim uma economia de 50% relativamente a um registo estereofónico dos mesmos sons graves. É o que se chama de estéreo junto (em Inglês, joint stereo).

### 3.2.4 MP3

O brevete original do algoritmo de compressão MP3 foi deposto em 1996 pelo instituto de pesquisa Alemão Fraunhofer, que faz parte do grupo Thomson Multimedia.

Desenvolvido no quadro de recomendações do Moving Picture Experts Group, ele tornou-se o algoritmo de compressão de som na norma MPEG e o seu verdadeiro nome é *MPEG-layer3*, o número final exprimindo o nível de compressão (1, 2 ou 3, este último número designando a taxa mais forte com uma razão de 10:1).

MP3 utiliza todas as técnicas descritas anteriormente, ao qual junta em primeiro uma codificação RLC (que permite evitar a codificação de dados sucessivos idênticos ou muito vizinhos) para terminar pela aplicação aos dados finais do algoritmo de Huffman.

Todas estas técnicas, com e sem perdas, permitem atingir taxas de compressão interessantes, e em acréscimo estas são adaptáveis em função da banda passante disponível para a difusão em tempo real (streaming) do ficheiro comprimido, isto é, em função do número de bits permitido pelo segundo (débito binário). Claro, que quanto menor for o débito binário, mais degradada é a qualidade. Assim, a 64 Kbits/s um minuto de música ocupa somente 469 KB para uma taxa de compressão de 22:1.

A título de comparação, o mesmo minuto de música não comprimido ocupa um pouco mais do de 10 MB sobre um CD de áudio. Mas a muito forte compressão do MP3 a 64 kbits/s é obtida à custa de uma qualidade medíocre, sendo todas as frequências compreendidas entre 11 e 20 kHz completamente suprimidas. Um débito binário de 96 Kbits/s melhora, evidentemente, um pouco

as coisas, mas é apenas a partir 128 Kbits/s e com uma taxa de compressão de 11:1 que nos aproximamos realmente da alta fidelidade. Finalmente a 192 Kbits/s, podemos considerar que a qualidade auditiva do CD de áudio é globalmente preservada; mas a taxa de compressão não ultrapassa 7:1, o que constitui apesar de tudo uma boa performance.

Ao MP3 clássico, o MP3 Pro apareceu em 2001 e juntou uma técnica específica de reconstrução da banda passante, muito gravemente amputada por um débito binário de 64 Kbits/s. Esta evolução do algoritmo responde aos desejos dos profissionais das telecomunicações que visam sempre as bandas passantes de mais baixas para uma qualidade aceitável.

MP3 Pro visa de facto uma melhoria notável de qualidade de reprodução musical a baixos débitos binários, atribuindo - sobre 64 Kbits/s disponíveis por convenção- 60 Kbits/s à codificação MP3 clássica(com uma banda passante limitada a 11 KHz) e 4Kbits/s a informações permitindo restabelecer parcialmente as altas frequências após a recepção do sinal. Esta técnica, denominada SBR (Spectral Bandwidth Replication) usa um codificador específico para a parte das altas frequências e permitindo atingir a banda passante a 16 KHz, o que melhora evidentemente a qualidade de restituição. No entanto, esta dupla codificação conduz a um aumento do tempo de descodificação e solicita muito mais do que o MP3 clássico a potência do processador. Ainda por cima, a melhoria da qualidade é apenas sensível para os débitos baixos, pois o algoritmo clássico é de facto satisfatório no que diz respeito à banda passante para débitos de 128 Kbits/s ou mais.

### 3.3 Formatos de ficheiros de áudio digital

De entre os muitos formatos de ficheiro de áudio propostos pelo mercado fiquemos pelos mais utilizados:

**WAV (Waveform Audio File Format)** é o formato proprietário do ambiente Windows sobre PC. Ele é lido directamente pelo leitor multimédia do Windows ou pelo leitor QuickTime sobre Mac OS assim que pela quase totalidade do software de edição ou de composição musical.

**AIF ou AIFF (Audio Interchange File Format)ou SND (SouND)** é o standard Mac, que é também reconhecido pelo software de PC (Sound Forge, Cool Edit pro). Ele tornou-se assim o primeiro formato de troca de várias plataformas.

**AU** formato desenvolvido pelo Unix, lido pela maior quase totalidade do software profissionais de tratamento de áudio

**Real Audio (RA, RM ou RAM)** formato proprietário adaptado aos débitos limitados da Internet para a difusão de som ou de vídeos em streaming. Estes ficheiros podem ser lidos graças a aplicações específicas RealPlayer.

**QuickTime (MOV ou QT)** destinado na sua origem ao ambiente Macintosh, hoje está disponível para PC e pode ser utilizado para realizar streaming.

**VOC** formato proprietário de Creative Labs (SoundBlaster).

**MP3** (*Mpeg – Layer3*) recomendado pelo MPEG, este formato apresenta o interesse maior de permitir um taxa de compressão importante sem alterar de forma notória a qualidade sonora, pelo menos até taxas de ordem 12:1. Numerosos leitores permitem a restituição de ficheiros .mp3: WinAmp, Nad mp3 Player, UnrealPlayer, JetAudio, etc.

**MID** trata-se aqui de um formato muito específico visto que os sons não são digitalizados como nos formatos WAV, AIF ou outros. A norma MIDI (Music Instrument Digital Interface) é de facto um protocolo material e de software permitindo a interligação e a comunicação de sintetizadores, amostradores, sequenciadores, de caixas de ritmos e claro de computadores. Codificado em linguagem MIDI, as informações relativas ao timbre do instrumento, à altura da nota, à potência, à sua duração, às nuances de interpretação, etc. (de facto tudo o que define o jogo de um músico...) pode ser transmitido ou registrado em tempo real e são compreendidos por todas as máquinas suportando esta norma. Um ficheiro MIDI restitui o som graças às informações codificadas seguindo uma norma, de onde o seu muito pequeno peso. Mas, consequência necessária, a qualidade desta restituição está directamente ligada à qualidade da máquina que interpreta o ficheiro, por exemplo, quando se trata de um computador, a carta de som. A norma geral MIDI foi criada para normalizar as referências à sonoridade dos diversos instrumentos.



# Capítulo 4

## Cor e Codificação da cor

Neste capítulo pretendemos desenvolver um entendimento dos vários modelos de cor e suas propriedades, de forma a podermos aplicar este conhecimento para uma correcta selecção de cores para visualização de imagens e vídeos.

### 4.1 A luz e a cor

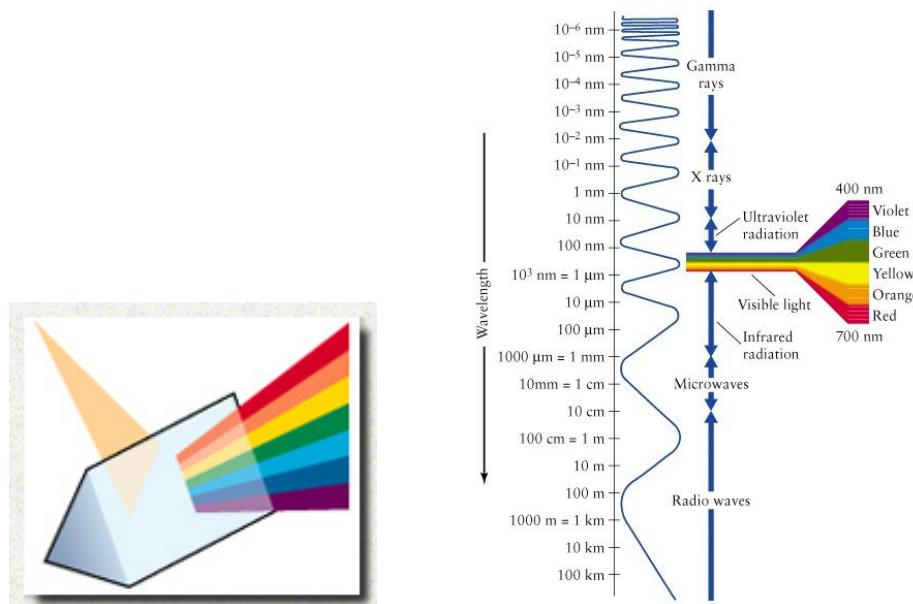


Figura 4.1: Espectro da luz.

A luz é uma onda electromagnética. A luz visível é uma onda electromagnética entre os 400nm e os 700 nm (onde nm representa nanómetros,  $10^{-9}$  metros). A sua cor é caracterizada pelo comprimento de onda da luz. A luz branca contém todas as cores do arco íris.

## 4.2 A percepção das cores

**1801** Thomas Young mostra que a percepção das cores pode ser explicada pela presença de três nervos na retina que são estimulados respectivamente pelo vermelho, pelo verde e pelo azul.

**1851** Hermann Ludwig Ferdinand von Helmholtz re-descobriu e desenvolveu a teoria de Young. Segundo ele é a síntese das três cores primárias que permite ao cérebro humano realizar todas as matizes de cores do espectro da natureza.

**Actualmente** Sabe-se que a retina integra não apenas terminações nervosas sensíveis às cores (entre 6 a 7 milhões de cones principalmente concentrados no centro da retina), mas também células especializadas na captura de bases luminosas repartidas sobre toda a superfície da retina, os bastonetes entre 115 a 120 milhões).

Os bastonetes são sensíveis à intensidade luminosa em toda a gama do comprimento de onda a que o olho humano é sensível. Assim, não conseguem descriminar entre luz recebida num comprimento de onda ou outro diferente, isto é, não detectam a cor.

Ao contrário dos bastonetes, os cones são sensíveis à luz apenas em certas gamas de comprimento de onda. assim existem cones sensíveis à luz na zona do vermelho ( $\rho$ ), na zona do verde ( $\gamma$ ) e na zona do azul ( $\beta$ ). Os cones necessitam de níveis de luminosidade mais elevados do que os bastonetes e, por esta razão, o olho humano não é capaz de detectar a cor dos objectos em condições de iluminação muito fraca como a noite.

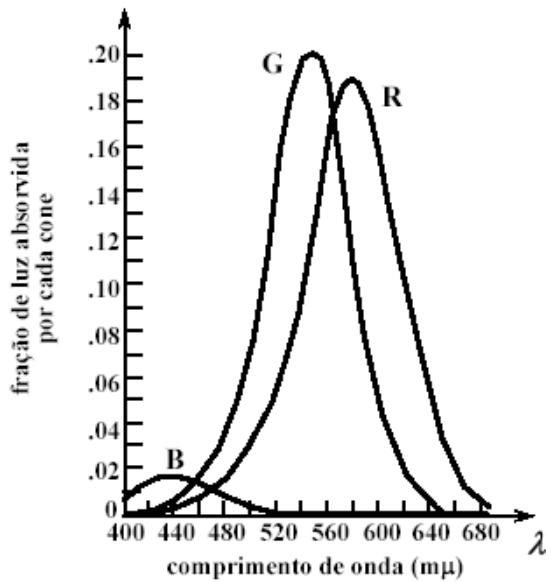


Figura 4.2: Luz absorvida.

A tabela 4.2 apresenta as gamas de comprimento de onda a que os três tipos de cones são sensíveis, o comprimento de onda em que cada um deles apresenta

Tipo de cone	Cor principal	Distribuição relativa (%)	Gama detectada (nm)	$\lambda$ de maior sensibilidade (nm)	Fracção de luz absorvida $\lambda_{max}$ (%)
$\beta$	azul	4	350-550	440	2
$\gamma$	verde	32	400-660	540	20
$\rho$	vermelho	64	400-700	580	19

Tabela 4.1: Características dos cones

maior sensibilidade, a sua distribuição relativa média e a fracção da luz incidente que absorve.

A tabela 4.2 e a figura 4.2 permitem-nos tirar algumas notas:

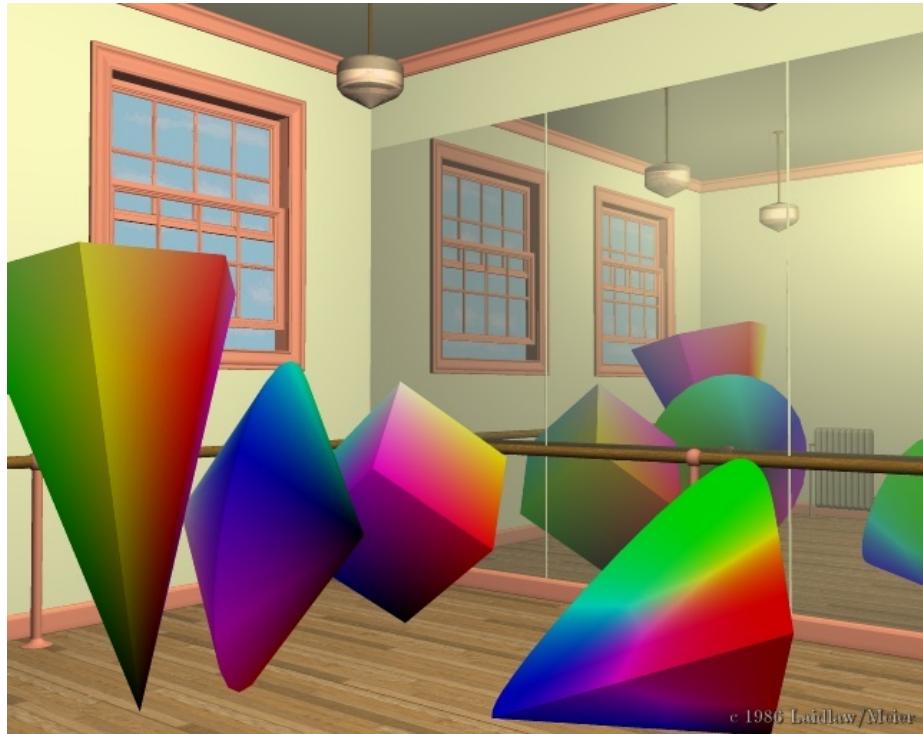
- Verifica-se que o tipo de cones do tipo  $\rho$  existentes na retina é quase o dobro do número de cones do tipo  $\gamma$ . O número de cones do tipo  $\beta$  é muito inferior ao número de cones de qualquer dos outros tipos.
- A maior sensibilidade do olho humano deverá verificar-se na gama de comprimentos de onda detectados pelos cones  $\gamma$  e  $\rho$ , a zona intermédia entre o verde e o vermelho (+- 550 nm - amarelo).
- É na gama de comprimento de onda mais baixa (zona do azul) onde existe menor sensibilidade à cor por parte de todos os tipos de cones.
- Os comprimentos de onda entre 400 e 660 são, em geral, detectados pelos três tipos de cones, mas cada tipo de cone detecta um dado comprimento de onda com uma sensibilidade diferente. Por exemplo, à luz com um comprimento de onda de 500 nm (cor ciano) correspondem sensibilidades dos cones  $\beta$  de 20%, dos cones  $\gamma$  de 30% e dos cones  $\rho$  de 10%. Para um comprimento de onda de 550 nm (amarelo), teremos respostas de 0% para  $\beta$ , de 99% para  $\gamma$  e de 80% para  $\rho$ .
- É a diferença das respostas destes três tipos de cones que permite interpretar diferentes comprimentos de onda como correspondendo a cores diferentes.

Como veremos na secção seguinte, alguns modelos de cor vão tirar proveito das características que acabamos de analisar.

### 4.3 Os modelos de cor

Numerosos modelos de representação dos espaços de cor utilizam uma decomposição sobre um plano representando a luminosidade e dois planos representando a informação da cor. Os significados e as definições destes três planos varia ligeiramente de um modelo ao outro, mas a sua interpretação mantém-se mais ao menos estável.

O plano da luminosidade permite representar a intensidade luminosa de um ponto independente da cor. Ele modela assim o funcionamento dos bastonetes do olho humano. Podemos assim trabalhar sobre a luminosidade de um ponto sem alterar a sua cor. Este plano corresponde à versão a preto e branco de uma imagem a cores. Foi realmente por razões de compatibilidade entre os ecrãs a cores e a preto e branco que apareceram estes tipos de modelos.



Inversamente, os dois planos das cores permitem representar o tom independentemente da luminosidade. Eles são normalmente construídos para explorar as propriedades de percepção do olho humano.

Para aplicações de compressão estes tipos de modelos pode ser vantajoso. Vários estudos mostraram que o nosso olho é muito mais sensível às informações de luminosidade do que às informações de tons. Assim, no processo de compressão com perdas podemos introduzir bastantes mais perdas nestes planos de cor do que no plano de luminosidade. Isto será explorado por formatos que utilizam as normas JPEG ou que trabalham com cores indexadas.

### 4.3.1 Modelo RGB

Este é o modelo mais conhecido de todos. É ele que é usado para as cores dos nossos ecrãs ou televisores. Cada cor é obtida por síntese aditiva das três cores primárias: vermelho, verde e azul.

**Síntese aditiva** O modo colorímetro de base para as aplicações numéricas é baseado sobre o funcionamento dos ecrãs de vídeo (computador, televisão, ...) que restituem os tons pela adição das três cores primárias: vermelho, verde e azul.

Neste modelo um tom é obtido através de uma combinação linear  $\theta = \rho \times R + \gamma \times G + \beta \times B$ , com  $(\rho, \gamma, \beta) \in [0, 1]^3$ .

Os triplos  $(0,0,0)$  e  $(1,1,1)$  correspondem respectivamente ao preto e ao branco.

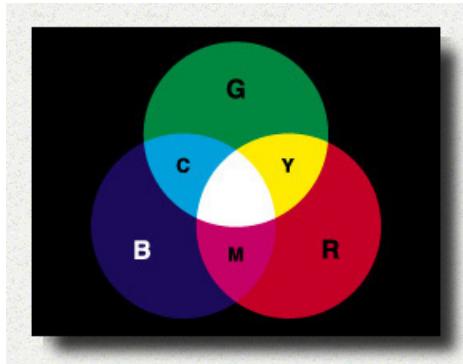


Figura 4.3: Cores primárias e secundárias no modelo de cor RGB.

Uma cor é definida pelo valor numérico atribuído a cada cor primária, isto é, por três inteiros. Se esses valores estão todos no máximo a cor resultante é o branco. Em contrapartida se os valores são todos nulos obtemos a cor preta. Se o valor das três cores primárias é idêntico obtemos cinzentos mais ou menos escuros dependendo da distância destes valores ao máximo.

Considerando uma base de um byte para cada cor podemos modelar  $2^8 \times 2^8 \times 2^8 = 2^{24} = 16777216$  cores. Uma imagem em modo 16 milhões de cores é conhecida por imagem em "true color". À vista do olho humano que é capaz de distinguir somente 350 000 cores diferentes este valor é mais do que suficiente.

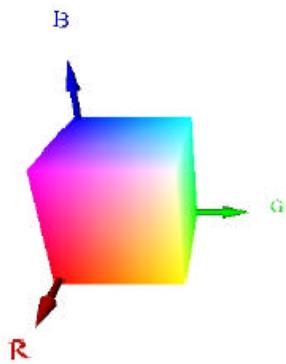


Figura 4.4: Representação do modelo de cor RGB.

O espaço de cor RGB pode ser representado por um cubo, (figura 4.4), onde as cores actuam como vectores. A origem comum dos vectores R, G e B constitui o ponto ou os valores onde os três pontos são iguais a 0 e que corresponde ao preto. O extremo vértice oposto do cubo corresponde ao ponto onde os valores das três coordenadas é máximo e por isso corresponde ao branco.

Sobre os vértices do cubo equidistantes de duas cores primárias encontram-se as cores secundárias: amarelo, magenta e ciano.

Este modelo é dependente dos periféricos utilizados na aquisição e na representação (devido às características individuais dos materiais).

### 4.3.2 modelo CMY

O modelo CMY (ciano, magenta e amarelo), é fundado sobre a capacidade de absorção da luz pela tinta de impressão depositada sobre o papel.

A síntese das cores é subtractiva: uma parte da luz branca que clareia o papel é absorvida, isto é, subtrai a luz reflectida pelos pigmentos coloridos de ciano, magenta e amarelo, absorvendo os seus complementares (ciano/vermelho, magenta/verde, amarelo/azul).

**Síntese subtractiva: CMY** Este tipo de síntese é próprio aos objectos que reflectem luz: para obter as matizes desejadas procede-se por subtração das cores reflectidas por uma superfície branca. Para isso usam-se filtros com as cores que queremos subtrair.

Seja uma fonte luminosa branca emitindo em direcção de três filtros ciano, magenta e amarelo. A cor ciano, sendo composta de verde e azul deixa passar estas duas cores, mas opõe-se à passagem do vermelho. A cor magenta, sendo composta de vermelho e de azul, deixa passar estas duas cores mas opõe-se à passagem do verde. Da mesma forma o filtro amarelo opõe-se à passagem da cor azul. A sobreposição dos três filtros não deixa passar qualquer das três cores primárias vermelho, verde e azul.

O espaço colorímetro subtractivo é adequado para as aplicações de impressão sobre suportes físicos.

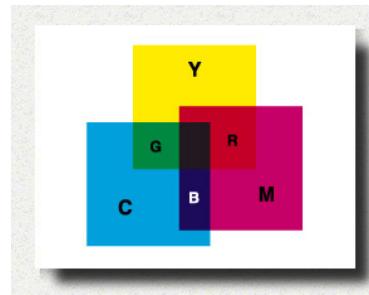


Figura 4.5: Cores primárias e secundárias no modelo de cor CMY.

Neste modelo, tal como no modelo RGB, cada cor é obtida modificando as proporções na mistura das cores primárias. Neste caso o valor máximo em todas as cores dá-nos o preto enquanto que estando as três cores nulas temos o branco.

#### Transformação RGB - CMY

$$\begin{bmatrix} c \\ m \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} r \\ g \\ b \end{bmatrix} \quad (4.1)$$

#### Transformação CMY - RGB

$$\begin{bmatrix} r \\ g \\ b \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} c \\ m \\ y \end{bmatrix} \quad (4.2)$$

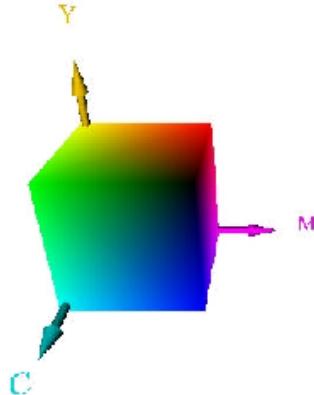


Figura 4.6: Representação do modelo de cor CMY.

#### 4.3.3 modelo CMYK

O espaço colorímetro subtractivo CMY é adequado para as aplicações de impressão sobre suportes físicos. Mas por razões técnicas ligadas às propriedades físicas das tintas de impressão, a sobreposição das três cores ciano, magenta e amarela, com saturações máximas, não é suficiente, geralmente, para absorver todo o espectro luminoso e o preto é assim difícil de produzir.

Por estas razões os sistemas de impressão usam geralmente uma decomposição que separa a informação do tom e da luminosidade. Um quarto plano é introduzido no modelo, correspondendo à cor preta. É claro que neste modelo chamado CMYK, os planos ciano, magenta e amarelo não são os mesmos do modelo CMY. Para formar estes quatro planos é necessário primeiro fazer uma conversão num espaço do tipo luminância/crominância ou tom/saturação e luminosidade, conforme veremos nas secções seguintes.

$$K = \min\{C, M, Y\}$$

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} \Rightarrow \begin{bmatrix} C - K \\ M - K \\ Y - K \end{bmatrix}$$

#### 4.3.4 Modelo HSV (Hue, Saturation, Value)

**H - Hue** Representa o tom. As diferenças no tom dependem das variações no comprimento de onda da luz que chega ao olho. O tom pode ser representado visualmente por um círculo de tom que vai do vermelho ao verde e volta ao vermelho.

**S - Saturation** S a saturação da cor. Esta é medida em termos de diferença entre uma cor com um cinzento com o mesmo nível de brilho. Quanto mais baixa for a saturação, mais cinzenta é a cor. Quando a saturação é zero a cor é cinzenta.

**V - Value** Representando a intensidade luminosa. O brilho é determinado pelo grau de reflectividade da superfície física que recebe a luz. Quanto maior for o brilho mais clara é a cor.

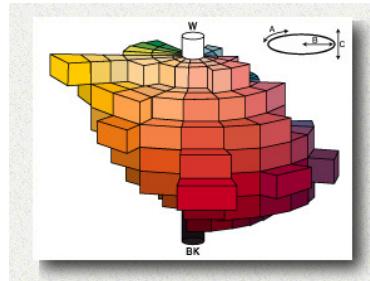


Figura 4.7: Representação do espaço HSV. A: Hue; B: Value; C: Saturation..

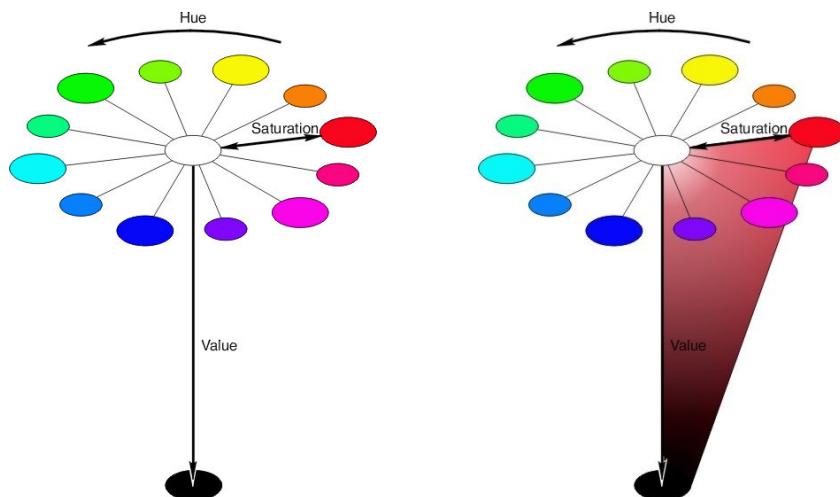


Figura 4.8: Representação do modelo HSV. Na esquerda com variações no Hue. Na direita com a saturação e o valor a variarem para um tom específico.

### 4.3.5 Modelos CIE

Os modelos de cor CIE (Comission Internationale de l'Eclairage) são uma família de modelos matemáticos que descrevem a cor em termos de tom (hue), brilho (value) e saturação (saturation). Os modelos de cor CIE incluem os modelos CIE XYZ, CIELAB, CIELUV. Vamos ver como funcionam alguns destes modelos.

#### Modelo CIE XYZ

O modelo CIE XYZ apareceu em 1931 e define um espaço de cor no qual se inscrevem todas as cores do espectro visível. Cada uma das cores é definida pela mistura em proporções específicas de três cores primárias X, Y e Z.

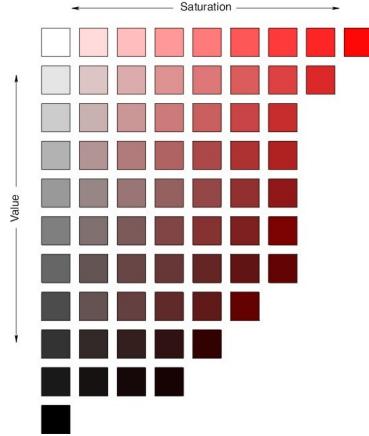


Figura 4.9: Variação da saturação e do valor para um tom vermelho específico.

Podemos obter um diagrama contendo todas as cores do espectro visível. Este sistema é construído de tal forma que valores iguais das três cores primárias produzem branco.

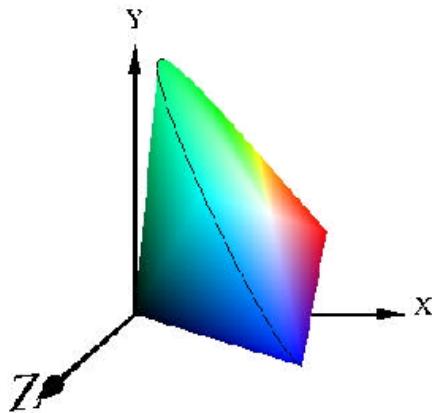


Figura 4.10: Modelo de cor CIE.

Normalizando os três valores XYZ podemos obter dois valores cromáticos:

$$x = \frac{X}{X + Y + Z} \quad y = \frac{Y}{X + Y + Z} \quad (4.3)$$

Claro que está definido outro valor cromático, mas este é redundante. Realmente, sabendo que  $x + y + z = 1$ , é suficiente saber dois valores para poder representar o terceiro como:  $z = 1 - x - y$ .

#### Transformação RGB - XYZ

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 0.49 & 0.31 & 0.20 \\ 0.17697 & 0.81240 & 0.01063 \\ 0 & 0.01 & 0.99 \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix} \quad (4.4)$$

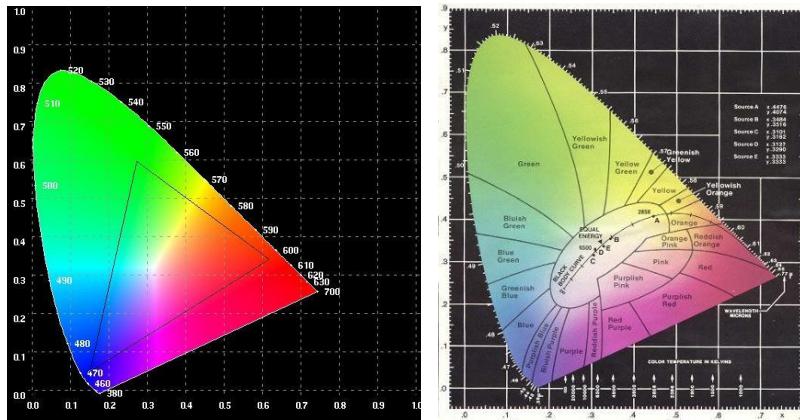
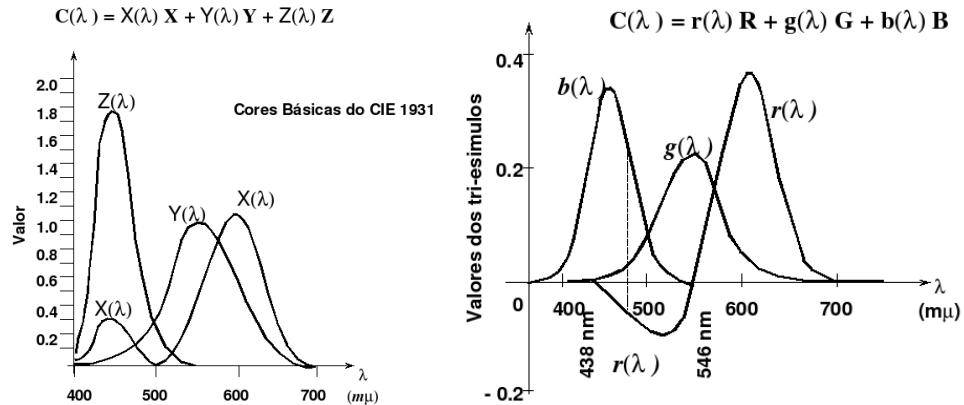


Figura 4.11: Modelo de cor CIE mapeado nas coordenadas X e Y.

Figura 4.12: Esquerda: *Color-matching functions* observadas no standard da CIE. Direita: *Color matching functions* do CIE 1931 RGB. Estas funções são a quantidade de primárias necessárias para obter a cor teste sobre as primárias.

### Modelo CIE xyY

Para caracterizar a luminância, CIE escolheu por convenção a coordenada Y: uma cor é sempre definida pelas suas coordenadas x, y e Y.

### Transformação XYZ - xyY

$$x = \frac{X}{X + Y + Z} \quad y = \frac{Y}{X + Y + Z} \quad Y = Y \quad (4.5)$$

### Transformação xyY - XYZ

$$X = \frac{x}{y} Y \quad Y = Y \quad Z = (1 - x - y) \frac{Y}{y} \quad (4.6)$$

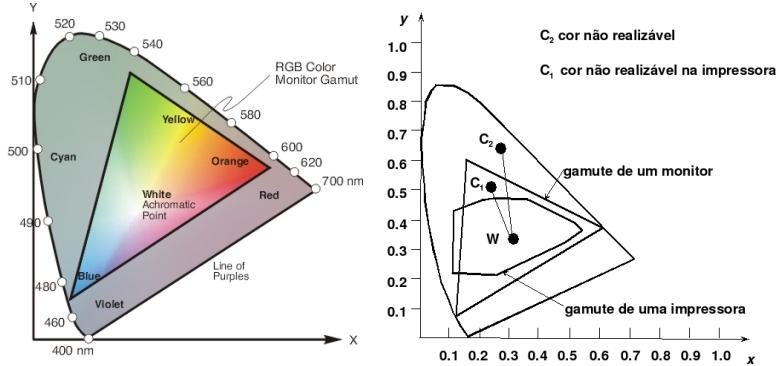


Figura 4.13: Esquerda: representação do modelo RGB sobre a representação do modelo XYZ. Direita: Representação das gamas de cor representáveis por um modelo de cor RGB e por um modelo de cor CMY.

#### Não uniformidade perceptual do espaço CIE XYZ

Tendo em conta as diferentes sensibilidades do olho humano às diferentes cores, o espaço CIE XYZ é percebido como não uniforme: diferenças numéricas de cor com alguma importância podem ser percebidas com dificuldade, enquanto outras diferenças muito inferiores podem ser apercebidas facilmente.

O sistema visual humano tem um maior poder discriminatório nos tons azuis do que nos tons verdes.

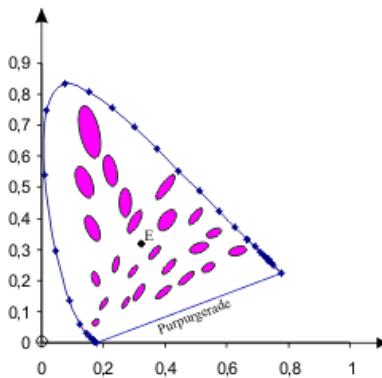


Figura 4.14: xyY.

#### Modelo CIE LAB

Como vimos atrás a percepção da cor não obedece a um mecanismo linear. A CIE tomou em conta esta não linearização da percepção visual humana propondo em 1960 uma escala uniforme de cor e depois estandardizando em 1976 a luminosidade ( $L^*$ ).

$$L^* = 903.3 \frac{Y}{Y_n}, \quad \text{se} \quad \frac{Y}{Y_n} \leq 0.008856$$

$$L* = 116 \frac{Y^{\frac{1}{3}}}{Y_n}, \quad \text{se} \quad \frac{Y}{Y_n} > 0.008856$$

onde  $Y_n$  designa a luminância de referência do branco.

Para  $Y = 0$  temos  $L* = 0$  e para  $Y = Y_n$  temos  $L* = 100$ . Assim a luminosidade varia entre 0 e 100. A diferença entre dois valores de luminosidade só é perceptível pelo olho humano quando superior ou igual a 1. Contrariamente à luminância, a luminosidade é percebida linearmente, o que significa que tem um comportamento uniforme no espaço das cores.

A partir deste parâmetro a CIE normalizou dois sistemas: CIE LUV e CIE LAB. O CIELUV é geralmente usado para calibrar monitores. Quanto ao espaço LAB ele apresenta o interesse de quantificar as cores segundo as classificações realizadas pelo pintor Albert H. Munsell e apresentadas no seu famoso atlas (figura 4.15). Este espaço é o mais uniforme dos espaços CIE.



Figura 4.15: Atlas de Albert H. Munsell.

O espaço LAB é usado por vários software de tratamento de imagem como espaço pivot para realização de conversão entre diversos espaços de cor. Isto deve-se ao facto dele ser um espaço independente dos dispositivos.

Neste espaço temos:

$$a* = 500 \left[ \left( \frac{X}{X_n} \right)^{\frac{1}{3}} - \left( \frac{Y}{Y_n} \right)^{\frac{1}{3}} \right] \quad (4.7)$$

$$b* = 200 \left[ \left( \frac{Y}{Y_n} \right)^{\frac{1}{3}} - \left( \frac{Z}{Z_n} \right)^{\frac{1}{3}} \right] \quad (4.8)$$

Resumindo, no espaço LAB (ou  $L^*a^*b^*$ ) três valores permitem representar as cores:

- $L^*$  designa a luminosidade expressa do 0 (preto) ao 100 (branco);
- $a^*$  exprime os valores da gama do verde ao vermelho, graduados de -100 a 100;
- $b^*$  exprime os valores da gama do azul ao amarelo, graduados de -100 a 100.

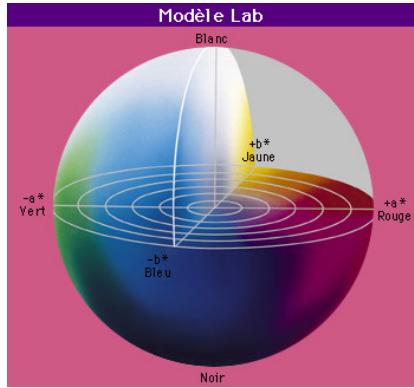


Figura 4.16: Representação do modelo CIE LAB.

#### 4.3.6 Modelos que separam luminância de crominância

Nestes modelos é usado um plano Y para a luminância e dois planos para a crominância. A luminância é a soma das três componentes básicas (R, G, B) moduladas segundo a sensibilidade da nossa percepção visual.

Vimos anteriormente que para energia igual é a luz verde que provoca maior sensação visual. A sensibilidade relativa das três cores é de 1 para o verde, 0.5 para o vermelho e 0.2 para o azul.

Para calcular a luminância a fórmula utilizada é a seguinte:

$$Y = 0.2999R + 0.58G + 0.114B.$$

Os coeficientes de ponderação de cada cor R, G e B são calculados usando a relação seguinte:

$$\text{Coeff}_{cor_X} = \frac{\text{sensibilidade}_{cor_X}}{\text{sensibilidade}_{cor_R} + \text{sensibilidade}_{cor_G} + \text{sensibilidade}_{cor_B}}$$

Por exemplo:

$$\text{Coeff}_{cor_R} = \frac{0.5}{1 + 0.5 + 0.2} = 0.2999$$

$$\text{Coeff}_{cor_G} = \frac{1}{1 + 0.5 + 0.2} = 0.58$$

$$\text{Coeff}_{cor_B} = \frac{0.2}{1 + 0.5 + 0.2} = 0.114$$

Para a crominância existem diferentes notações consoante os standard:

- NTSC (National Television System Committee) - 1953, Americano.

$$I = 0.74(R - Y) - 0.27(B - Y)$$

$$Q = 0.48(R - Y) - 0.49(B - Y).$$

- PAL (Phase Alternation by Line) - 1967 Alemão.

$$U = 0.147R - 0.289G + 0.436B = 0.493(B - Y)$$

$$V = 0.615R - 0.515G - 0.100B = 0.877(R - Y).$$

- SECAM (Séquentiel Cleur À Mémoire) - 1967 Francês.

$$\begin{aligned} D_r &= -1.9(R - Y) \\ D_b &= 1.5(B - Y). \end{aligned}$$

- CCIR (Comité Consultatif International des Radiocommunications) (para imagem fixa)

$$\begin{aligned} C_r &= 128 + 128 \frac{1}{0.701}(R - Y) \\ C_b &= 128 + 128 \frac{1}{0.886}(B - Y). \end{aligned}$$

O olho é mais sensível à luminância do que à crominância. Este facto é usado na compressão com perdas para reduzir o débito binário sem alterar a percepção das imagens.

#### 4.4 Gamma e modo R'G'B'

A luminância produzida por um tubo catódico está evidentemente ligada à tensão do sinal que ele recebe em entrada. Mas esta relação não é linear: uma mesma variação do sinal gera de facto uma pequena variação de luminância nas tensões baixas e uma grande variação de luminância nas tensões elevadas.

Para representar esta não linearidade da reprodução da luminância utiliza-se um parâmetro numérico denominado gamma ( $\gamma$ ). Gamma é o expoente da função potência que aproxima esta não linearidade:  $L = v^\gamma$ , onde  $v$  designa o sinal de entrada.

A esta não linearidade é preciso juntar uma não linearidade na percepção humana da luminância. Neste caso a não linearidade ocorre ao inverso da não linearidade dos ecrãs catódicos. Assim, para a mesma variação objectiva da luminância, a variação que nós percebemos é elevada nos níveis baixos e fraca nos níveis elevados. Para exprimir esta não linearidade perceptual da luminância usamos uma função potência com expoente sub-unitário ( $1/\gamma$ ).

Assim, todos os dispositivos de aquisição vão utilizar uma correção gamma para modificar os valores de entrada de acordo com a nossa percepção. O expoente de correção é neste caso  $1/\gamma$  ou gamma de codificação, enquanto na altura de representar uma imagem no monitor é feita a correção  $\gamma$  ou gamma de descodificação.

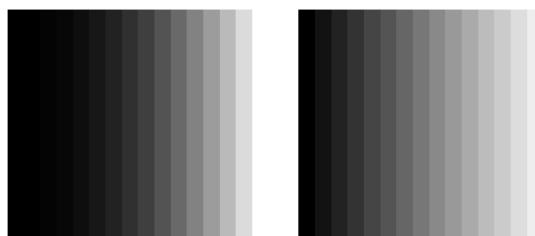


Figura 4.17: Sem correção Gamma (esquerda); Com correção Gamma (direita).

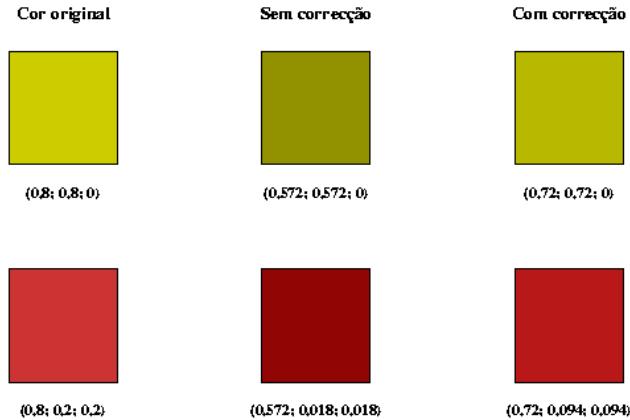


Figura 4.18: Cores representadas sem e com a correcção gamma..

## 4.5 Paletes de cor e cor indexada

Hoje em dia, a maior parte das imagens presentes nas aplicações multimédia são realizadas em modo RGB (16 milhões de cores). No entanto, alguns utilizadores dispõe ainda de computadores configurados para representar apenas um número de cores muito inferior.

Chamamos paletes de cores ao sub-conjunto das gamas disponíveis por um sistema particular. Assim, uma magnifica imagem original RGB em milhões de cores terá de ser traduzida utilizando somente as cores disponíveis na paleta do sistema de visualização.

As cores da imagem original que não estão disponíveis na paleta de cores do sistema terão de ser aproximadas. Para suavizar o efeito de redução de número de cores de uma imagem "full-color" representada numa paleta de cores do sistema é frequentemente, usada uma técnica de "dithering" que consiste em usar cores diferentes em pixels contíguos para criar a ilusão de uma outra cor. Pode também ser usada esta técnica para em imagens a preto e branco dar a impressão de existirem nuances de cinzento.

A indexação da cor é a técnica que relaciona, ou indexa, cada cor da imagem com uma cor da paleta do sistema. Vamos ver como se processa esta indexação.

Comecemos por notar que para guardar 256 valores são necessários 8 bits. Se estes valores representarem cores, como é que podemos dividir os 8 bits pelos três cores primárias?

Se fizermos por exemplo 3 bits para o R, 3 para o G e 2 para o B teremos um número limitados de matizes em cada primária (8 para o R, 8 para o G e 4 para o B). Logo esta não é a melhor solução.

Uma solução para este problema está em definir 256 cores cada uma delas codificada sobre 8 bits por cor primária. Assim qualquer da cores ditas "true color" pode ser retida. De seguida cada cor seleccionada é designada por um índice numa tabela de 256 cores. Assim, uma imagem em 256 cores é chamada em cor indexada, visto que o seu valor representa o índice da paleta aonde está definida a cor.

Numa paleta cada parâmetro é codificado numa tabela da paleta sobre um número de bits idêntico ao utilizado na gama das cores. Para uma paleta de 256

valores, é assim necessário uma tabela de dimensão  $256 \times 24$ . Dispomos assim de uma paleta de 256 cores sobre os 16 milhões de cores disponíveis.

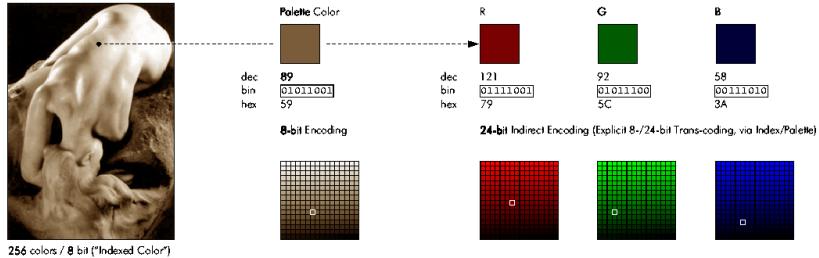


Figura 4.19: Imagem representada usando uma paleta exacta.

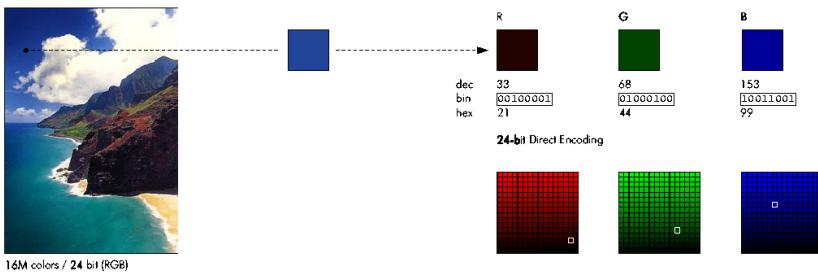


Figura 4.20: Imagem em cor indexada.

Para alterar o modo de representação de uma imagem em "true color" em cores indexadas, os software de tratamento de imagem propõe vários tipos de paletes:

- paleta exacta: disponível unicamente se a imagem RGB de origem não utiliza mais do que 256 cores; na paleta serão registadas as cores exactas do original;
- paleta sistema: paleta por defeito de um sistema de exploração; as paletes do sistema Windows e Macintosh são diferentes;
- paleta uniforme: obtidas por tomada uniforme de amostras de cores no cubo RGB; o número total de cores obtidas será igual ao cubo do número de tomas (com cinco níveis de tomas, por exemplo, obtemos  $5^3 = 125$  cores);
- paleta perceptiva: favorece, entre as cores presentes na imagem, aquelas às quais o olho humano é mais sensível;
- paleta selectiva: similar à paleta perceptiva valorizando as zonas de cores mais importantes e preservando as cores web (conforme a paleta web), o que garante a melhor fidelidade de reprodução;
- paleta adaptativa: favorece os tons dominantes na imagem; uma imagem RGB na qual dominam as matizes de verde e de azul (o que é normal-

mente o caso das paisagens) conduzirá à criação de uma paleta composta principalmente de verde e azul;

- palete web: constituída de 216 cores, ela é utilizada pelos navegadores web, qualquer que seja a plataforma, para representar as imagens sobre um monitor limitado a 256 cores. Para obrigar o navegador a apresentar sempre as mesmas 216 cores, constitui-se uma paleta por tomadas regulares de 7 níveis(seis intervalos) para cada cor primária. Cada cor do modelo RGB pode tomar valores entre 0 e 255. Os valores 0 e 255 são conservados para cada cor primária, o que permite obter o preto, o branco assim como todas as cores primárias (RGB) e secundárias (CMY).

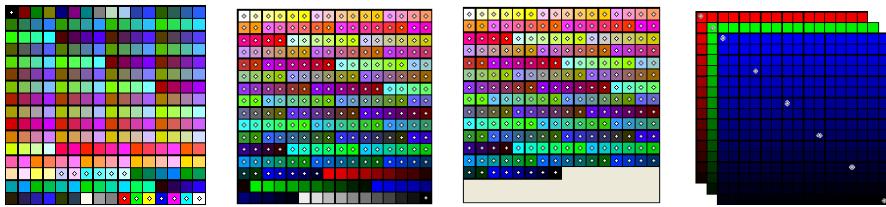


Figura 4.21: Da esquerda para a direita: paleta do Windows; paleta do Macintosh; paleta do WEB; 3 Canais RGB.

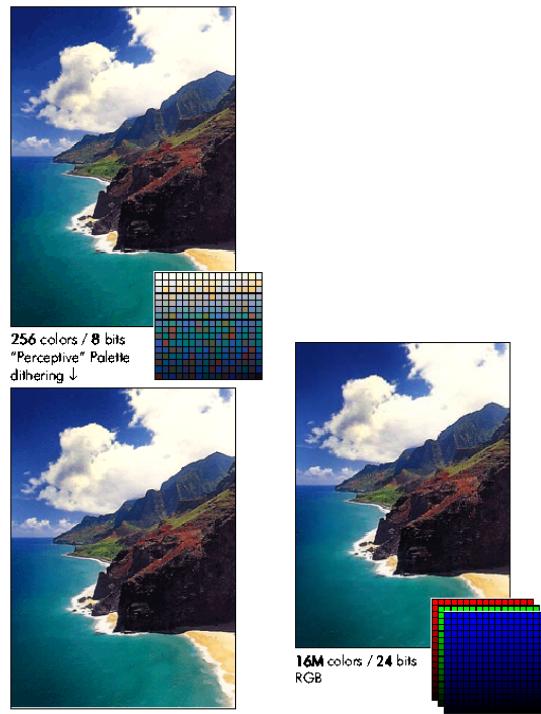


Figura 4.22: Representação de uma imagem em usando diferentes tipos de paletes.



Figura 4.23: Representação de uma imagem em usando diferentes tipos de paletes.

## 4.6 Conclusões

A representação numérica da cor é um tema muito vasto do qual acabamos de ver uma pequena parte.

## 4.7 Problemas

1. Uma fotografia de  $10 \times 20$  polegadas foi digitalizada tomando 300 amostras por polegadas (300 dpi) e usando 24 bits por amostra.
  - (a) Imagine que a fotografia anterior continha apenas 32 cores diferentes. Indique uma técnica que poderá ser usada para diminuir o número de bits por amostra, mas permitindo continuar a representar a cor em RGB. Indique quantos KB precisará neste caso para guardar a imagem (explique todos os cálculos efectuados).
  - (b) Imagine que a fotografia digitalizada anteriormente em cor verdadeira (RGB) foi transformada em YUV. Explique o que representa o plano Y;
  - (c) Qual o espaço necessário para gravar a mesma fotografia em 32 níveis de cinzento; Explique o processo que usa para fazer essa passagem.
2. Indique três espaços de cor diferentes. Explique a base de funcionamento do espaço RGB. Indique aplicações que usem cada um desses três espaços.
3. Uma fotografia de  $10 \times 20$  polegadas foi digitalizada tomando 300 amostras por polegadas (300 dpi) e usando 24 bits por amostra (true color). A mesma fotografia foi digitalizada tomando 600 amostras por polegada e usando 8 bits por amostra (cor indexada).
  - (a) Qual o espaço ocupado pelas diferentes imagens digitalizadas, em KB.
  - (b) Que diferenças poderão ser percebidas entre as duas imagens digitalizadas.
  - (c) Será possível ter as duas imagens digitalizadas com exactamente as mesmas cores? Explique.
  - (d) Podemos representar uma imagem usando diferentes espaços de côr. Explique a razão de representar uma imagem em diferentes espaços de côr.
  - (e) Explique como funciona e para que serve a cor indexada.



# Capítulo 5

## Imagen Digital

Podemos distinguir duas categorias de imagens segundo o tipo de dados que são codificados e depois guardados para permitir a reconstrução da imagem. As imagens em mapa de pontos nas quais o ficheiro é constituído pelos dados da cor de cada ponto, e as imagens vectoriais para as quais os ficheiros contêm a descrição matemática das figuras elementares que os constituem.

### 5.1 Imagen em mapa de pontos

As imagens em mapa de pontos são constituídas por um conjunto de pontos alinhados horizontal e verticalmente como linhas e colunas de uma matriz. Um pixel é o elemento mais pequeno da imagem ao qual se associa individualmente uma cor (ou nível de cinzento).

Todas as imagens reproduzidas num ecrã são imagens em mapa de pontos, e o mesmo acontece para as imagens obtidas através de um scanner ou de um aparelho fotográfico numérico.

Podemos dividir as imagens em mapa de pontos em várias categorias:

- imagens em dois níveis (ou preto e branco), com 1 bpp;
- imagem em níveis de cinzento, com 8 bpp;
- imagens em cor indexada, com 8 bpp;
- imagens hicolor, com 16 bpp; Nestes temos duas variantes: na primeira temos 5 bits por cada uma das cores primárias e 1 bit para opacidade ( $2^{15} = 32768$  cores); na segunda variante temos 5 bits para o canal R, 6 bits para o canal G e 5 bits para o canal B ( $2^{16} = 65536$ ).
- imagens "true color", com 24 bpp; encontramos imagens "true color" com 32 bpp. Nestas o quarto byte é usado para a componente alpha que representa o nível de opacidade: 0 para transparência total e 255 para opacidade total.

#### 5.1.1 Tamanho das imagens

Sabendo a largura e a altura, em pixéis, de uma imagem e a profundidade de quantificação podemos calcular o tamanho do ficheiro de uma imagem em mapa

de pontos (não comprimido) como:

$$T(KB) = \frac{\text{largura (pixel)} \times \text{altura (pixel)} \times \text{profundidade (bits/pixel)}}{8(\text{bits/byte}) \times 1024(\text{bytes/KB})} \quad (5.1)$$

Por exemplo uma imagem com  $640 \times 480$  e com profundidade 24, terá tamanho:

$$T(KB) = \frac{640 \times 480 \times 24}{8 \times 1024} = 900KB. \quad (5.2)$$

### 5.1.2 Resolução da imagem

Chamamos resolução de uma imagem ao número de pontos digitalizados, afixados ou imprimidos por unidade de largura. Quanto maior é o número de pontos maior é a resolução da imagem e vice-versa.

A resolução é medida em pixel por polegada (ppp).

Com que resolução devemos digitalizar uma imagem? Tudo depende da utilização final dessa imagem. Se desejamos integrar a imagem numa página web, isto é, representá-la num monitor, estaremos interessados na resolução do monitor. Por outro lado, se pretendemos imprimi-la, estaremos interessados na resolução de impressão.

### Entrelaçamento

As imagens em mapa de pontos podem permitir uma visualização progressiva através da técnica de entrelaçamento. Vamos ver de seguida para que serve e como funciona esta técnica.

Mesmo com a redução de tamanho permitida pela indexação de cores uma imagem pode demorar um tempo considerável até poder ser visualizada.

**Exemplo** Uma imagem constituída por 360 linhas e 270 colunas, num total de 97.200 pixels, mais  $256 \times 3 = 768$  para guardar o mapa de cores levaria cerca de 45,9 segundos a ser transmitida, quando transmitida usando uma linha de grande velocidade (19200 bps). O utilizador teria que esperar todo este tempo até poder observar a imagem e ficar com uma ideia do seu conteúdo.

Muita da informação de uma imagem é redundante, isto é, não é necessário visualizar todos os pixels de uma imagem para formar uma ideia do seu conteúdo, principalmente quando se procura uma imagem. Foi exactamente para permitir a observação do conteúdo de uma imagem sem dispor de todos os seus pixels que foi desenvolvida a técnica do entrelaçamento a partir de uma técnica semelhante empregue na transmissão de imagens de televisão.

A técnica básica de transmissão parcial e progressiva de imagens por entrelaçamento consiste em reordenar as linhas das imagens, organizando-as em vários grupos. Cada grupo contém parte das linhas da imagem total e, se uma linha for atribuída a um grupo, essa linha não fará parte de qualquer outro grupo.

A transmissão da imagem grupo a grupo permite que o utilizador comece a formar uma ideia da imagem após apenas algumas linhas terem sido transmitidas. O utilizador começa por ver uma imagem nos seus traços gerais, com

poucas linhas e sem detalhes, e, à medida que vão sendo apresentados novos grupos de linhas, verificará um aumento progressivo da qualidade e do número de detalhes. Esta característica pode ainda hoje ser observada durante o carregamento de imagens transmitidas pela WWW quando o fluxo de dados é baixo.

### 5.1.3 Qualidades e defeitos das imagens em mapa de pontos

#### Vantagens

- São capazes de produzir graduações de nuances e de cores muito finas. Constituem o suporte electrónico ideal de imagens com tons contínuos como fotografias ou imagens criadas com software de desenho.
- São constituídas por pixeis, o que permite o tratamento ponto a ponto sem modificar os objectos ou as formas, e são perfeitamente adaptadas a uma representação sobre ecrã ou impressão;
- São directamente guardadas na memória, logo são representadas muito rapidamente no ecrã (muito mais depressa do que as imagens vectoriais que, como veremos de seguida, devem ser reconstituídas).

#### Desvantagens

- A qualidade destas imagens está directamente dependente ao material de aquisição e de reprodução (por exemplo a resolução do ecrã, da impressora, do scanner);
- O facto de serem imagens de pontos exige uma grande quantidade de espaço em memória (grande quantidade de pontos cujas características de cor são definidas individualmente).

### 5.1.4 Formatos de imagem em mapa de pontos

Temos os formatos proprietários (ligados a um software específico), não proprietários e temos ainda os meta-formatos.

Os meta-formatos, tal como o nome indica, não são formatos propriamente ditos, mas são envelopes que garantem uma descrição dos documentos que eles contêm (quer se trate de imagem em mapa de pontos, vectorial, de texto ou mesmo uma combinação dos três). Esta descrição sendo independente das plataformas e dos softwares usados para a sua criação, permite aos meta-formatos ser lidos por todas as máquinas disposto do "plug-in" adequado. Os principais meta-formatos são: PDF, PS, EPS, sendo a sua origem desenvolvida para o texto e a imagem vectorial, eles serão descritos na parte consagrada à imagem vectorial.

Visto que existe uma grande quantidade de formatos dedicados à imagem em mapa de pontos, apresentaremos aqui apenas as características dos formatos mais utilizados: GIF, JPEG, PNG.

### Formato GIF (Graphic Interchange Format)

- Este formato foi concebido em 1980 pela Compuserve para necessidades específicas de edição em linha;
- Utiliza compressão sem perdas LZW; Esta procura motivos que se repitam ao longo de um eixo horizontal, e cada vez que encontra uma nova cor aumenta o tamanho do ficheiro. Assim, é mais eficaz quando aplicada a imagens que contém poucas modificações aquando de uma passagem horizontal;
- Profundidade de pixel não superior a 8 bits (logo no máximo 256 cores);
- GIF é optimizado para compressão de imagens contendo poucas cores diferentes e apresentando grandes quantidades de pixéis da mesma cor (logos, esquemas, diagramas a preto e branco, etc);
- É adaptado a imagens com forte contraste e a texto mas não à fotografia realista;
- A variante GIF87a aceita entrelaçamento e a versão 89a junta ainda a possibilidade de transparência e animação;
- Em modo GIF entrelaçado, as imagens afixadas primeiro em forma de blocos, tornam-se progressivamente mais e mais nítidas. O entrelaçamento não afecta o tamanho nem a velocidade de afixação, é destinado a ganhar tempo, pois o visitante pode ficar rapidamente com uma ideia global da imagem e decidir, ou esperar para a visualização definitiva da imagem ou interromper o carregamento da imagem e prosseguir a navegação.

### Entrelaçamento no formato GIF

O formato de imagem GIF emprega entrelaçamento em 4 passagens pelas linhas das imagens. Numerando as linhas de 0 a n, a primeira passagem processa as linhas 0, 8, 16, etc., a segunda passagem as linhas 4, 12, 20, etc., e a terceira passagem as linhas 2, 6, 20, 24, etc. Finalmente, a quarta passagem processa todas as linhas que ainda não foram processadas (1, 3, 5, 7, etc.).

Cada uma das primeiras duas passagens processa assim 1/8 da imagem, a terceira 1/4 e a quarta passagem a metade restante das linhas.

---

**Exemplo** Para o exemplo apresentado anteriormente de uma imagem de  $360 \times 270$  pixéis e com 256 cores, todas as linhas do primeiro grupo serão visíveis ao fim de 6.1 segundos, as do segundo grupo ao fim de 11.7 segundos e as do terceiro grupo ao fim de 23.1 segundos. A imagem ficará completa com o quarto grupo ao fim de 45,9 segundos.

---

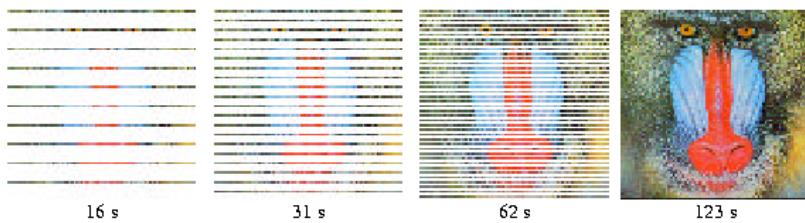


Figura 5.1: Visualização da imagem baboon em formato gif entrelaçado.

#### **Formato JPEG (Joint Photographic Experts Group)**

- Standard de compressão para imagens fixas;
- Utiliza algoritmos de compressão com perdas, com taxa de compressão variável;
- Permite adaptar a qualidade da imagem ao tipo de aplicação pretendida;
- Suporta imagens codificadas em 24 bpp ("true color");
- Constitui o formato web para fotografia realista;
- Permite uma representação sequencial, linha a linha ou progressiva, por passagens sucessivas;
- A compressão JPEG é excelente em zonas contendo variações subtils de cor, mas menos bom a comprimir zonas de cor uniforme;
- Aceita entrelaçamento, na sua versão progressiva, mas não aceita transparência nem animação;
- O JPEG2000 é um sucessor deste. Ambos estes algoritmos serão analisados no capítulo da compressão de imagens com perdas.

#### **Formato PNG (Portable Network Graphics)**

- Formato mais recente que reúne as principais qualidades dos seus predecessores, eliminando a maioria dos defeitos;
- Utiliza compressão sem perdas;
- Permite codificações até 48 bpp;
- Atinge níveis de compressão próximos dos JPEG (mas sem perdas);
- Permite entrelaçamento (representação da imagem em 7 passagens);
- Permite transparência por canal alpha;
- Este formato é livre (não utiliza qualquer algoritmo de domínio privado);
- Este formato é simples, apresenta grande portabilidade e compressão eficiente;

- Ainda não é tão usado na Internet como o GIF pois os antigos navegadores ainda não o reconhecem;
- O seu familiar MNG (Multiple Image Network Graphics) é utilizado para imagens PNG animadas. Apesar de apresentar muitas vantagens (transparência, fidelidade de cores, utilização de pistas para combinar animações diferentes, gestão da sincronização e da difusão das animações por scripts externos) necessita de tempo de descompressão importante e por isso não é muito usado.

### Entrelaçamento no formato PNG

No PNG o entrelaçamento processa pixels individuais em lugar de processar linhas. Os pixels a transmitir em cada passagem são determinados por um padrão de  $n \times n$  pixels de que são feitas tantas cópias quanto as necessárias para cobrir toda a imagem, estando as cópias justapostas. O algoritmo de entrelaçamento Adam7, empregue pelo formato de imagem PNG, define um padrão de  $8 \times 8$  pixels

```

1 6 4 6 2 6 4 6
7 7 7 7 7 7 7 7
5 6 5 6 5 6 5 6
7 7 7 7 7 7 7 7
3 6 4 6 3 6 4 6
7 7 7 7 7 7 7 7
5 6 5 6 5 6 5 6
7 7 7 7 7 7 7 7

```

que aplica a toda a imagem, começando pelo canto superior esquerdo. Se, por exemplo, a imagem a transmitir apresentar a dimensão de  $16 \times 16$  pixels, cada uma das 7 passagens transmitirá novas linhas cujo número e comprimento variarão com a ordem da passagem, tal como a tabela 5.1 apresenta.

**Exemplo** Retomando o exemplo anterior, as 7 passagens do algoritmo de entrelaçamento Adam7 encontrar-se-ão completas ao fim de 1.1s, 1.8s, 3.2s, 6.1s, 11.8s, 23.1s e 45.9s, respectivamente.

### Outros formatos

- TIFF (Tagged Image File Format) - Formato de ficheiro de imagem referenciada. Foi desenvolvida para manipular imagens de alta densidade na altura da troca de ficheiros entre aplicações ou plataformas informáticas. Usa compressão LZW, e preserva a qualidade da imagem original. Verdadeiro standard para publicação assistida por computador (PAO) este formato de imagem em mapa de pontos é tomado em conta pela grande parte de aplicações gráficas (desenho, formatação, edição). A maioria de scanners de gabinete produzem imagens TIFF, o mesmo acontece com os aparelhos fotográficos numéricos de qualidade;

Passagem número	Número de linhas transmitidas	Comprimento das linhas	Bytes transmitidos
1	2	2	4
2	2	2	4
3	2	4	8
4	4	4	16
5	4	8	32
6	8	8	64
7	8	16	128
		Total	256

Tabela 5.1: Número e comprimento das linhas transmitidas de uma imagem de  $16 \times 16$  pixeis pelo algoritmo de entrelaçamento Adam7 (formato PNG)

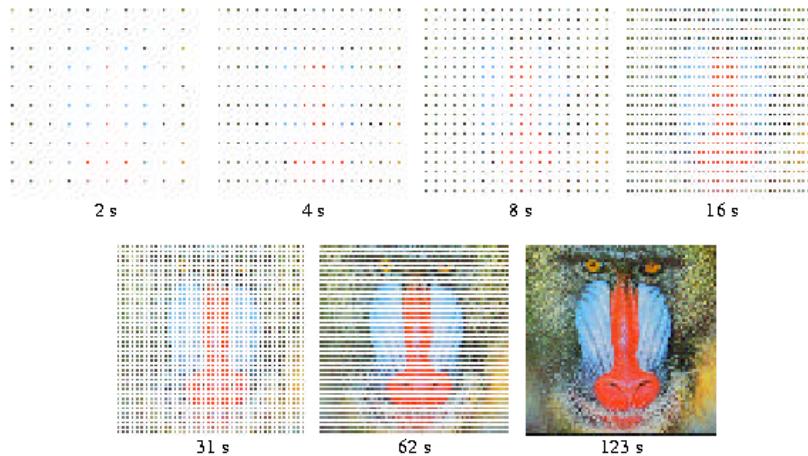


Figura 5.2: Visualização da imagem babbon em formato PNG entrelaçado.

- PICT - Desenvolvido pela Apple em 1984, é aceite por todos os softwares gráficos em versão Mac OS e é particularmente eficaz para compressão de imagens contendo vastas zonas uniformes. A versão original só aceitava 8 cores, mas as versões mais recentes permitem 16 ou 32 bpp. Assim, toma em conta imagens em níveis de cinzento, imagens em cores indexadas, e imagens RGB;
- FIF (Fractal Image File) - é um formato de ficheiros de imagens obtidas por compressão Fractal;
- DIB (Device Independent Bitmap) - também designado por BMP, é o formato standard em mapa de pontos utilizado pelo Windows. Para assegurar uma independência total do material, este formato junta uma tabela de cores que redefine para cada cor um valor RGB. Cada ficheiro DIB inclui dois cabeçalhos que detalham as características da imagem (resolução horizontal e vertical, esquema de compactação, tamanho, número de cores,

etc.);

- WBMP (Wireless BitMaP) - é um formato de imagem optimizado para os periféricos móveis (telefone, PDA, i-Mode). Este formato não aceita profundidade de pixel superior a 1 bit, por isso só pode tomar em conta imagens a preto e branco. Efeitos de trama permitem criar pseudo-níveis de cinzento que melhoram consideravelmente o aspecto da imagem;
- RAW - significa bruto, não trabalhado. Em fotografia numérica um ficheiro RAW contém os dados originais tais como são captados pelos CCD. O formato RAW assegura uma profundidade de pixel superior aos 24 bits de formato TIFF ou JPEG (30 a 36 bits para certos aparelhos de alta gama). Ideal para os profissionais da imagem numérica.

Formato	Compressão	Codificação	Progressivo	Transparência	Animação
BMP	Não	24 bits	Não	Não	Não
TIFF	Não	32 bits	Não	Não	Não
GIF	Sem perda	1 a 8 bits	Sim (entrelaçado)	Sim (GIF89a)	SIM (GIF89a)
JPEG	Com perdas	24 bits	Sim	Não	Não
PNG	Sem perdas	8 a 48 bits	Sim	Sim	Não

Tabela 5.2: Características de diferentes formatos

## 5.2 Imagem vectorial

A imagem vectorial é constituída por um conjunto de figuras elementares descrita por dados matemáticos. Os ficheiros de criação ou armazenamento de uma imagem vectorial descrevem as diferentes figuras como objectos gráficos independentes uns dos outros. Assim esses objectos podem ser manipulados e transformados independentemente.

### 5.2.1 Qualidades e defeitos das imagens vectoriais

#### Vantagens

- As informações são descritas textualmente logo eficazmente comprimidas;
- O tamanho da memória é independente do tamanho da imagem;
- É possível aplicar facilmente e sem perda de precisão transformações geométricas: deslocamentos, translações, alteração de tamanho, rotações,...;
- Os diferentes objectos podem ser manipulados e transformados independentemente e com grande precisão;
- São independentes dos periféricos e da resolução, assim são automaticamente colocadas na escala para ser imprimidas de forma precisa sobre qualquer periférico de saída.

### Desvantagens

- O tamanho do ficheiro varia em função da complexidade da imagem;
- Não podemos usar o formato vectorial para descrever uma imagem muito complexa, por exemplo fotografia;
- O tempo de representação de uma imagem vectorial é superior em relação a uma imagem em mapa de pontos (aumenta com a complexidade da imagem);
- Qualquer perda ou corrupção no ficheiro leva à perda da imagem na totalidade;

### 5.2.2 Formatos da imagem vectorial

Formatos Proprietários	Meta-formatos	Formatos para a Web
Adobe Illustrator (.ai)	Postscript (.ps)	Shockwave Flash (.swf)
Corel Draw (.cdr)	Encapsulated Postscript (.eps)	Scalable Vector Graphics (.svg)
Windows MetaFile (.wmf)	Portable Document Format (.pdf)	

Tabela 5.3: Formatos de imagem vectorial.

### 5.2.3 SVG: o HTML do vectorial

Em 2001, a W3C (World Wide Web Consortium) propôs a recomendação SVG (Scalable Vector Graphic) como linguagem de descrição de gráficos vectoriais bidimensionais para a Web. A SVG tem a vantagem de ser não proprietário e de ter funcionalidades que são fácil e rapidamente assimiláveis e exportáveis. A SVG é implementável em qualquer sistema de exploração ou periférico (desde o telefone portátil ao ecrã táctil).

A linguagem de descrição SVG permite criar gráficos 2D susceptíveis de conter não apenas formas vectoriais, mas também imagens em mapa de pontos e texto. Um ficheiro SVG contém linhas de texto (ASCII) que podem ser lidas pelo utilizador e facilmente interpretáveis por um software de leitura adequado (por exemplo um plug-in para navegador Internet).

A recomendação SVG define o vocabulário da linguagem SVG e também as regras que se aplicam à determinação do conteúdo dos elementos, à associação entre os elementos e seus atributos, aos conteúdos textuais, assim como as regras de declaração dos formatos de dados utilizados nos ficheiros anexos. Como toda a linguagem necessita de regras sintácticas, o W3C escolheu para o SVG a sintaxe XML (eXtended Markup Language) que é um derivado na norma internacional SGML (Standard Generalized Markup Language).

A linguagem XML é uma linguagem do tipo do HTML, mas ao contrário do HTML que foi concebido para representar os dados preocupando-se exclusivamente da sua aparência ou formato, o XML foi concebido para descrever os dados focando-se na sua estrutura.

Para criar um ficheiro SVG qualquer editor de texto serve. Encontramos porém alguns softwares concebidos para a criação do SVG:

- Amaya - <http://www.w3c.org> (open source W3C);
  - Sodipodi - <http://www.sodipodi.com> (GNU/Linux);
  - Inkscape - <http://www.inkscape.org> (open source).
- 

#### **Exemplo** Exemplo de um ficheiro SVG

```
<svg id="drapeau France" viewBox="0 0 300 200">
<title> Test SVG </title>
<g id="France" stroke ="blue">
    <path d="M 50 50 h 50 v 100 h -50 z" fill="blue"/>
    <path d="M 100 50 h 50 v 100 H 100 z" fill="white"/>
    <path d="m 150 50 H 200 V 150 H 150 v -100" fill="red"/>
</g>
</svg>
```

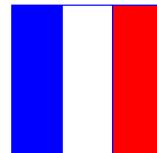


Figura 5.3: Imagem vectorial.

---

#### 5.2.4 Meta-formatos

**PS** PostScript é uma linguagem de descrição de páginas desenvolvida pela Adobe (1985), que garante a independência das páginas a imprimir em relação às características do sistema de impressão. Trata-se de uma verdadeira linguagem de programação. O script de um ficheiro PostScript consiste numa descrição do documento em formato ASCII, meticuloso e detalhada. Este script é gerado por um software. O formato PostScript nível 2 gera igualmente um certo número de técnicas de compressão (LZW, CCITT, JPEG).

**EPS** Este formato permite transferir os objectos codificados em PostScript entre diversas aplicações. Aceita os grafismos vectoriais ou em mapa de pontos. Aceita os espaços de cor LAB, CMYK, RGB , as imagens indexadas, níveis de cinzento. Não toma em conta o canal alpha (transparência).

**PDF** Criado pela Adobe para substituir o PostScript na edição profissional. Garante a descrição de gráficos vectoriais e de em mapa de pontos. As páginas PDF são idênticas às páginas PostScript, mas propõe funções de pesquisa e de consulta electrónica dos documentos. Os ficheiros PDF podem conter por exemplo links de hypertexto e índices electrónicos

### 5.3 Problemas

1. Indique 3 vantagens e 3 desvantagens das imagens bitmap em relação ás imagens vectoriais.
2. Explique para que serve e como funciona a técnica de entrelaçamento.
3. Explique que técnica poderemos usar para termos uma vizualização progressiva de uma imagem durante a sua descodificação.
4. Comente a afirmação: "Uma imagem vectorial ocupa sempre menos espaço em memória que uma imagem bitmap".



# Capítulo 6

## Codificação Fonte

O objectivo da compressão é de reduzir ao máximo o volume de armazenamento e de transferência de informação, mantendo no entanto a capacidade de restituir a integridade dessa informação. Pelo menos quando trabalhamos em compressão sem perdas.

Como reduzir o volume de dados digitais sem degradar a informação ou, pelo menos, só a degradando de forma imperceptível e controlável?

Constatando que a quase totalidade de dados que tratamos - imagem, texto, som - apresentam uma distribuição não uniforme de símbolos ou de sequências de símbolos.

**Texto** Os caracteres que utilizamos não apresentam a mesma probabilidade de aparecimento. Todo o texto apresenta uma estrutura interna forte, determinada pelas regras sintáticas, semânticas e gramaticais. Regras implicam regularidade e consequentemente formas repetitivas e redundância.

**Som** Se analisarmos um trecho de música, constatamos rapidamente que a distribuição das probabilidades de ocorrência de sons não é mais uniforme que a das letras num texto. Também eles seguem regras ou leis de construção.

**Imagen** Quando manipulamos imagens constatamos evidentemente que elas apresentam regularidades: longas séries de pixéis idênticos, ou quase idênticos: no céu azul, nas nuvens brancas, na relva verde; estruturas repetitivas; composição geral da imagem, etc. Tratam-se neste caso de redundância espaciais, i.e., motivos repetitivos no mesmo espaço.

**Vídeo** Logo que utilizamos sequências de imagens, as redundância temporais vêm juntar-se às redundância espaciais. De uma imagem à seguinte, e numa ordem de 25 imagens por segundo, muitos dos elementos mantém-se idênticos. Por exemplo, numa entrevista televisiva, uma grande parte da imagem (o fundo) mantém-se idêntica durante vários segundos.

São estas redundância espaciais e temporais que permitem a compressão dos dados, isto é, a remoção de dados inúteis. Para tirar partido destas redundância podemos usar algoritmos mais ao menos complexos, fundados sobre técnicas matemáticas quase sempre sofisticadas (compressão JPEG, MPEG, Wavelets, fractais, etc.).

## 6.1 Redundância e quantificação da redundância

Vamos aqui ver como podemos medir o potencial de compressão de que dispomos para uma ou outra série de dados. Para isso vamos ver que existe uma relação entre incerteza e informação. Assim, se conseguimos quantificar a informação conseguimos quantificar a redundância de informação e consequentemente os limites de compressão.

### 6.1.1 Incerteza vs. informação

A informação é condicionada pela variabilidade. Descrever o estado actual de um sistema que só pode conter um estado não comporta nenhuma informação. Informação e incerteza estão relacionados. Vamos ver como.

---

**Exemplo** Imaginemos a seguinte situação:

*Sejam três urnas contendo seis bolas cada. Na primeira estão seis bolas brancas, na segunda três bolas brancas e três bolas pretas e na terceira duas bolas brancas, duas bolas pretas e duas bolas vermelhas.*

Sendo o número de bolas nas três urnas o mesmo, a quantidade de informação em cada uma será a mesma?

Claro que não. Uma bola branca não nos dá qualquer informação se for tirada da primeira urna pois ela só contém bolas brancas: não há nenhuma incerteza quanto ao resultado de tiragem desta urna, logo nenhuma informação.

Em contrapartida, a incerteza é maior quando se trata de tirar uma bola branca da terceira urna (1 hipótese em 3) do que de tirar uma bola branca da segunda urna (1 hipótese em 2). Podemos dizer que a quantidade de informação é maior quando se trata de tirar uma bola branca da terceira urna.

---

O exemplo acima leva-nos a várias notas:

- uma experiência fornece informação se e só se o resultado dessa experiência elimina uma certa incerteza; uma experiência cujo resultado é conhecido a priori não fornece qualquer informação.
- a quantidade de informação e o nível de incerteza são directamente proporcionais: indicar o estado presente de um sistema susceptível de tomar um grande número de estados significa fornecer uma grande quantidade de informação.
- quando a informação vem substituir a incerteza, podemos utilizar para determinar a quantidade de informação a mesma medida que usamos para o grau de incerteza. Aumentar a informação relativa a um sistema é diminuir na mesma medida a incerteza: a informação varia no sentido inverso da incerteza.

### 6.1.2 Entropia

Em 1948, Claude Shannon enunciou uma fórmula matemática que permite calcular a quantidade média de informação de uma experiência tendo em conta o nível de incerteza próprio a essa experiência.

Seja uma experiência  $X$  susceptível de tomar os estados  $x_1, x_2, \dots, x_N$  com probabilidades  $p_1, p_2, \dots, p_N$ , respectivamente. Consideramos que as probabilidades  $p_i$  não variam ao longo do tempo e que se trata de um sistema completo de acontecimentos.

$$\sum_{i=1}^N p_i = 1 \text{ com } 0 \leq p_i \leq 1 \quad (6.1)$$

Nestas condições, podemos apresentar a experiência  $X$  sob a forma de uma variável aleatória:

$$x = \begin{bmatrix} x_1 & x_2 & \dots & x_n \\ p_1 & p_2 & \dots & p_n \end{bmatrix} \quad (6.2)$$

A incerteza de uma experiência  $X$  depende essencialmente da probabilidade da ocorrência dos estados possíveis do sistema.

Claude Shannon chamou  $H$  à entropia do conjunto das probabilidades  $p_1, p_2, \dots, p_n$ :

$$H(p_1, p_2, \dots, p_n) = \sum_{i=1}^N p_i \times \log_2 \left( \frac{1}{p_i} \right) = - \sum_{i=1}^N p_i \times \log_2(p_i) \quad (6.3)$$

Se os estados possíveis são equiprováveis, isto é, se eles têm todos a mesma probabilidade ( $p_i = 1/N$ , para  $i = 1, 2, \dots, N$ ), a equação 6.3 pode-se escrever como:

$$H(p_1, p_2, \dots, p_n) = \sum_{i=1}^N \frac{1}{N} \times \log_2(N) = \log_2(N) \quad (6.4)$$

**Exemplo** Seja uma imagem em níveis de cinzento com profundidade de pixel de 8 bits. Os valores representáveis com 8 bits vão de 0 a 255.

Suponhamos agora duas situações diferentes:

**Primeira situação** Todos os valores entre 0 e 255 são equiprováveis. Neste caso,  $p_i = 1/256$  para todo o  $i$  entre 0 e 255. De acordo com a equação da entropia

$$H(1/256, 1/256, \dots, 1/256) = \log_2(256) = 8.$$

Assim, não poderemos comprimir esta imagem pois o seu nível de redundância é nulo. Segundo a fórmula de Shannon a representação de cada símbolo exige 8 bits que é exactamente o número de bits usado originalmente.

**Segunda situação** O valor 128, por exemplo, tem uma probabilidade  $p_{128} = 0.99$  e todos os outros valores tem uma probabilidade constante  $p_i = 0.01/255$ . aplicando a fórmula de Shannon temos:

$$H(p_1, p_2, \dots, p_n) = \sum_{i=1}^N p_i \times \log_2\left(\frac{1}{p_i}\right) = -\sum_{i=1}^N p_i \times \log_2(p_i) = 0.16$$

Assim, a redundância absoluta é de  $8 - 0.16 = 7.84$  bits/valor ou, em valor relativo de  $7.84/8 = 0.98 = 98\%$ . A taxa de compressão destes dados é então de 50 : 1 (pois  $8/0.16 = 50$ ).

---

A fórmula de Shannon permite avaliar a taxa de compressão sem perdas de uma série de dados representados por um conjunto de símbolos (ou alfabeto). A entropia  $H$  exprime o número médio de bits por símbolo necessários para a codificação ideal de um determinado alfabeto.

Veremos mais tarde que as probabilidades de ocorrência dos símbolos serão exploradas por diferentes algoritmos de compressão sem perdas, por exemplo, a codificação entrópica de Huffman, Shannon-Fano, RLC, etc..

## 6.2 Os parâmetros da compressão

**Taxa de compressão** O parâmetro chave de um algoritmo de compressão é a taxa de compressão, a qual define a sua performance. Este é calculado como a relação entre o tamanho do ficheiro original ( $F_o$ ) e o tamanho do ficheiro comprimido ( $F_c$ ). Utilizamos geralmente para representar esta relação uma noção do tipo  $N : 1$  que indica que o volume dos dados de origem foi dividido por  $N$ . Por exemplo 10 : 1 significa que o tamanho do ficheiro original foi dividido por 10.

**Simetria e assimetria** Outros parâmetros que caracterizam igualmente um algoritmos de compressão são a sua velocidade e a qualidade de restituição do documento original.

Um sistema de compressão supõe um algoritmo de compressão e um de descompressão (codec). Estes algoritmos apresentam normalmente uma certa assimetria no que diz respeito ao tempo e à qualidade de compressão.

A velocidade, que define o tempo necessário ao processo de compressão / descompressão, pode ser simétrico quando o tempo necessário à compressão é equivalente ao tempo necessário à descompressão. Para a maior parte dos médias digitais, a velocidade é assimétrica: o tempo de compressão é mais longo do que o tempo de descompressão (para gravar em CDROM, aplicações de vídeo a pedido, etc.). Notemos igualmente que velocidades assimétricas autorizam em geral um melhor débito, ou taxa de compressão. No caso de aplicações multimédia em tempo real (vídeo-conferencia, streaming de áudio e vídeo em tempo real, etc) temos necessidade de compressores rápidos. O tempo de compressão é nestes casos quase igual ao tempo de descompressão. Em geral, esta velocidade simétrica reduz a taxa de compressão. Assim, simetria e assimetria podem influenciar a qualidade de compressão.

**Com ou sem perdas** Os métodos de compressão podem dividir-se naqueles que implicam perdas de informação (lossy) e naqueles que permitem encontrar a informação na integralidade após o processo de descompressão

(lossless). No primeiro caso o algoritmo de compressão é irreversível, pois ele não permite a partir do ficheiro descomprimido ( $F_d$ ), encontrar na integralidade o ficheiro original ( $F_o \neq F_d$ ). No segundo caso o algoritmo de compressão é reversível ( $F_d = F_o$ ).

## 6.3 Técnicas de base

### 6.3.1 Run Length Coding (RLC)

O método RLC consiste em recuperar as sequências repetitivas de caracteres e substitui-las por um caractere de controlo seguido do número de repetições. Assim. "aaaa...." transforma-se em #Na, onde:

- # indica o início de uma repetição;
  - N indica o número de repetições;
  - a é a sequência repetida.
- 

**Exemplo** A sequência "AAAABQEEEEEFFFFF" quando codificada usando 8 bits por símbolo requer um total de 14 bytes. Usando a codificação RLC ficará "#4ABQ#4E#4F" o que requer 11 bytes.

---



---

**Exemplo** A sequência "0000000001111000000" quando codificada usando 8 bits por símbolo requer um total de 19 bytes. Usando a codificação RLC ficará "#90#41#60" o que requer 9 bytes. Podemos até imaginar para este caso uma codificação mais eficiente usando apenas 2 bytes (16 bits): 080146. Neste caso estou a considerar que o primeiro bit diz-me como começa a sequencia e que os restantes valores são codificados em 3 bits (daí que os 9 zeros tiveram de ser codificados em 8 mais 1, isto é, na sequência 801).

---

#### Considerações finais sobre RLC

Constatamos facilmente que este método só convém aos ficheiros que contém um grande número de repetições de 4 caracteres ou mais.

Este método, simples, pode-se mostrar muito eficiente para codificar por exemplo imagens monocromáticas (por exemplo o caso de documentos transmitidos por fax onde encontramos longas sequências de pixels de preto e de pixels de branco) ou de imagens constituídas de grandes blocos da mesma cor.

Este método também pode ser usado considerando um ficheiro como um conjunto de bits. Repare que neste caso, desde que começemos por indicar qual o primeiro bit do ficheiro, resta-nos a indicar os valores das repetições.

### 6.3.2 Codificação relativa

Na codificação de longas séries de valores próximos, isto é, contendo pequenas variações de amplitude, podemos constatar que os primeiros bits de cada byte são próximos. Por exemplo, na altura da apresentação das temperaturas para fins meteorológicos ou industriais, podemos obter longas sequências de valores cuja variação só afecta as décimas ou as centésimas: obtemos assim bytes do tipo: 01011100 01011111 01011101 01011110 etc.

Utilizamos nestes casos um método de compactação por contagem:

- um caractere especial indica o início de uma codificação relativa (#);
- um número precisa de seguida o número de bits idênticos;
- de seguida escrevem-se esses bits;
- indica-se o número de bytes afectados por essa codificação;
- e finalmente os bits não repetidos desses bytes.

#### Exemplo

**010111000101111101011110101011110 ↔ #6010111400110110**

## 6.4 Algoritmos de codificação estatística

Como já vimos atrás, há, na quase-totalidade dos dados que tratamos, uma ordem subjacente que se traduz por redundância. Isto significa que há certos dados que ocorrem mais frequentemente do que outros. Assim, calcular as probabilidades de ocorrência dos objectos manipulados (bits, bytes, pixéis, etc. ) fornece uma informação capital sobre a sua repartição uniforme (todas as ocorrências são equiprováveis) ou não uniformes (alguns objectos têm uma probabilidade de ocorrência mais ou menos elevada).

Se a repartição é uniforme, a entropia é máxima: na ausência de redundância, todos os símbolos serão codificados com o mesmo número de bits (codificação de comprimento fixo).

Se a repartição não é uniforme, a entropia não é maxima, o que deixa entender, como veremos de seguida, que existe uma possibilidade de optimizar a codificação.

Os algoritmos que se seguem utilizam uma análise estatística ou entrópica que melhora consideravelmente a sua eficiência.

### 6.4.1 Método de Huffman

Neste método pretende-se:

- Codificar com o menor número de bits os símbolos em que a frequência de ocorrência é a maior.

- Os primeiros elementos de uma palavra código (o prefix) não pode contuir uma outra palavra de código (código prefixo). Isto permite-nos garantir uma descodificação instantânea.
- 

**Exemplo** Considere a sequênciа

*ABRACADABRA.*

Se codificarmos esta sequênciа usando 8 bits por cada símbolo necessitaremos de um total de 88 bits (8 bits x 11 símbolos = 88 bits). Considerando os dois pontos acima podemos imaginar uma codificaçao mais eficiente. Por exemplo:

$A- > 5- > 0$

$B- > 2- > 10$

$R- > 2- > 110$

$C- > 1- > 1110$

$D- > 1- > 1111$

Com estes códigos, a mensagem seria codificada como

01011001110011110101100.

Verificamos que neste caso usamos apenas 23 bits em vez dos 88 bits anteriores.

---

### Algoritmo

O algoritmo de Huffman consiste em partir de uma análise estatística do conteúdo do ficheiro, atribuir aos símbolos mais frequentes os códigos binários o mais curto possível, reservando os códigos mais longos aos símbolos que ocorrem menos vezes.

Este tipo de codificaçao obriga à criação de tabelas de frequências que terão de ser transmitidas visto que são necessárias para a descodificaçao.

Para aplicar a um ficheiro o algoritmos de Huffman temos de percorrer as seguintes etapas:

**Etapa 1** Leitura do ficheiro para cálculo da frequência de ocorrência de cada símbolo.

**Etapa 2** Classificaçao dos símbolos em função da sua frequência de ocorrência.

**Etapa 3** Reagrupamento sequêncial dos pares de símbolos de menor frequência, voltando a executar a classificaçao, se necessário.

**Etapa 4** Atribuição de um código a cada símbolo: atribui-se um 0 ou um 1 a cada elemento antes de regrupá-lo; O código de cada símbolo será constituído pela sequência (lida da direita para a esquerda) de 0 e 1 que se encontram no caminho de regrupamento do símbolo.

**Etapa 5** Codificação final.

---

**Exemplo** Vamos codificar a sequência:

*AMOREMOREORERE*

O alfabeto será constituído por seis símbolos: A, E, M, O, R, Espaço. Para comprimir esta sequência de dados usando o algoritmo de Huffman vamos seguir cada uma das etapas acima descritas.

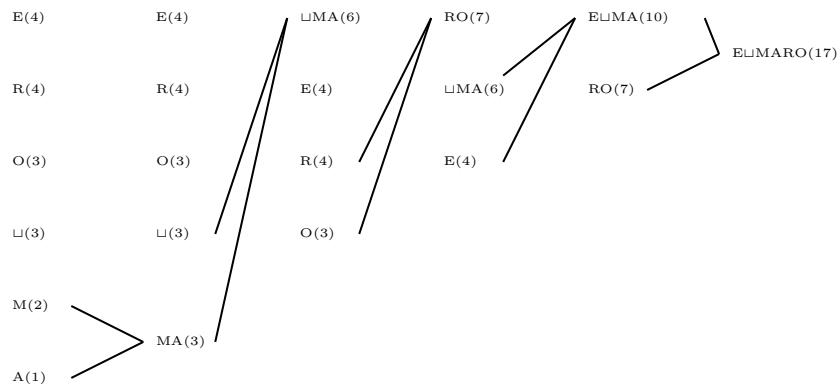
**Etapa 1** Establecer a frequência de ocorrência de cada símbolo.

$$f(A) = 1; f(E) = 4; f(M) = 2; f(O) = 3; f(R) = 4; f(\text{Espaço}) = 3$$

**Etapa 2** Classificação dos símbolos por ordem decrescente de frequência.

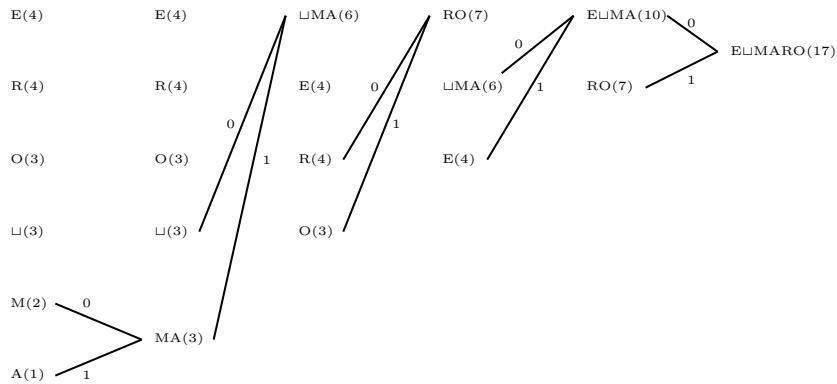
Símbolo $i$	Frequência $f_i$
E	4
R	4
O	3
Espaço( $\sqcup$ )	3
M	2
A	1

**Etapa3** Agrupamento das duas frequências mais pequenas, voltando a executar a classificação anterior, se necessário.

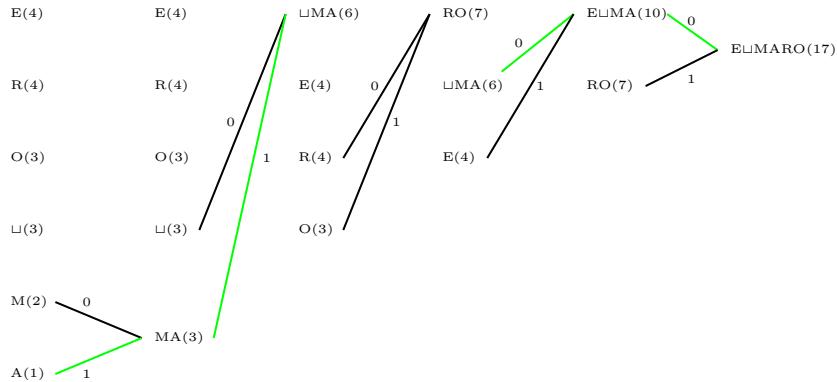


**Etapa4** Atribuição do código.

Para encontrar o código de cada símbolo, identificamos o seu caminho da direita para a esquerda, e recuperamos os 0 ou os 1 que se encontram pelo caminho.



Por exemplo, para o código do elemento A



Temos assim 0011 como código do elemento A. De forma similar recuperamos os restantes códigos.

Símbolo $i$	Código
E	01
R	10
O	11
Espaço	000
M	0010
A	0011

**Etapa 5** Compressão da cadeia:

*AMOREMOREORERE*

---

0011001011100100000101110010001110010001001

---

**Nota** Os códigos obtidos não são únicos.

**Nota** O código de prefixo obtido com o algoritmo de Huffman é otimal, visto que não existe nenhum outro código de prefixo cuja largura média seja inferior.

### Largura média

Sabendo a probabilidade de cada elemento do alfabeto com que estou a trabalhar, podemos facilmente calcular o número médio de bits que necessito para codificar uma determinada fonte. Para isso basta-me calcular a largura média dos códigos. Também o tamanho final da sequência codificada é simples de calcular, basta multiplicar a largura média pelo tamanho da minha fonte.

A largura média dos códigos é calculada por

$$l_{media} = \sum_i p_i l_i \quad (6.5)$$

sendo  $l_i$  a largura do código  $i$  (a largura ideal de um símbolo é dada por  $-\log_2(p_i)$ ).

---

**Exemplo** Temos no caso do nosso exemplo

$$\begin{aligned} l_{media} &= \frac{4}{17} \times 2 + \frac{4}{17} \times 2 + \frac{3}{17} \times 2 + \frac{3}{17} \times 3 + \frac{2}{17} \times 4 + \frac{1}{17} \times 4 \\ &= \frac{43}{17} = 2.529. \end{aligned}$$

$$\begin{aligned} H &= 2 \times \frac{4}{17} \log_2\left(\frac{4}{17}\right) + 2 \times \frac{3}{17} \log_2\left(\frac{3}{17}\right) + \frac{2}{17} \log_2\left(\frac{2}{17}\right) + \frac{1}{17} \log_2\left(\frac{1}{17}\right) \\ &= 2.47. \end{aligned}$$

Repare que o código de Huffman diz-se óptimo na medida em que é dos códigos de prefixo o que mais se aproxima do valor da entropia.

A mensagem codificada ainda guarda alguma redundância. Neste caso dizemos que o código tem uma redundância de

$$l_{media} - H = 2.529 - 2.47 = 0.059.$$


---

### Codificação de Huffman usando árvores binárias

A codificação de Huffman também pode ser executada usando árvores binárias. Será idêntica à apresentada anteriormente, mas desta vez em vez de iterarmos da esquerda para a direita iteramos de baixo para cima. Partimos com todos os símbolos como folhas da árvore e vamos juntando até chegarmos à raiz da árvore. Neste caso, os códigos recuperam-se partindo da raiz e descendo até às folhas.

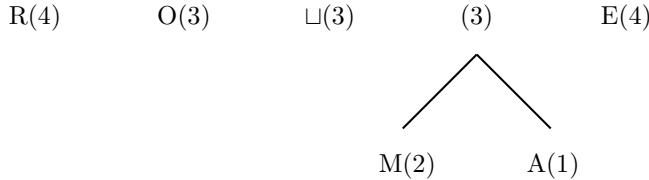
---

**Exemplo** O exemplo apresentado em cima ficaria da seguinte forma:

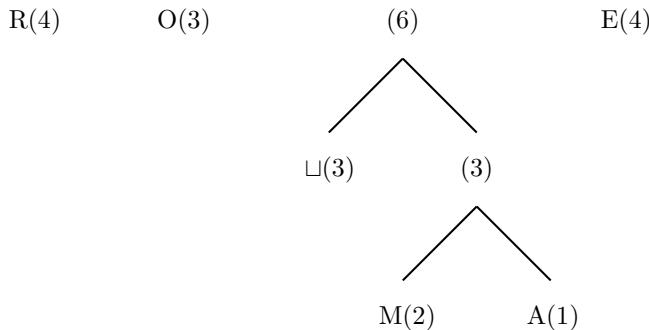
Começamos com todos os elementos como nodos da árvore.

R(4)      O(3)       $\sqcup(3)$       M(2)      A(1)      E(4)

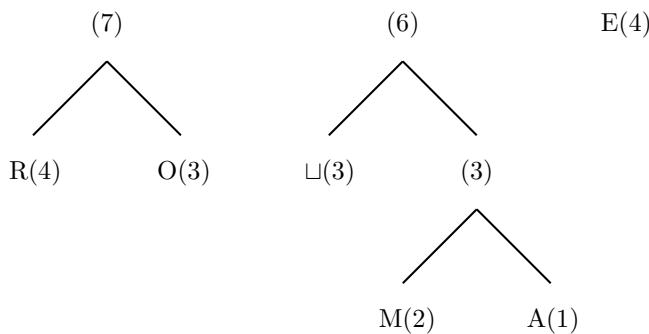
Juntamos os dois nodos com menor frequência. Neste caso o nodo com o elemento M e o nodo com o elemento A. Estes dois juntos dão origem a um nodo com frequência 3.



Continuamos a juntar os nodos com as menores frequências. Neste caso temos três nodos com frequência 3. Escolhemos dois deles e juntamos num nodo com frequência 6.

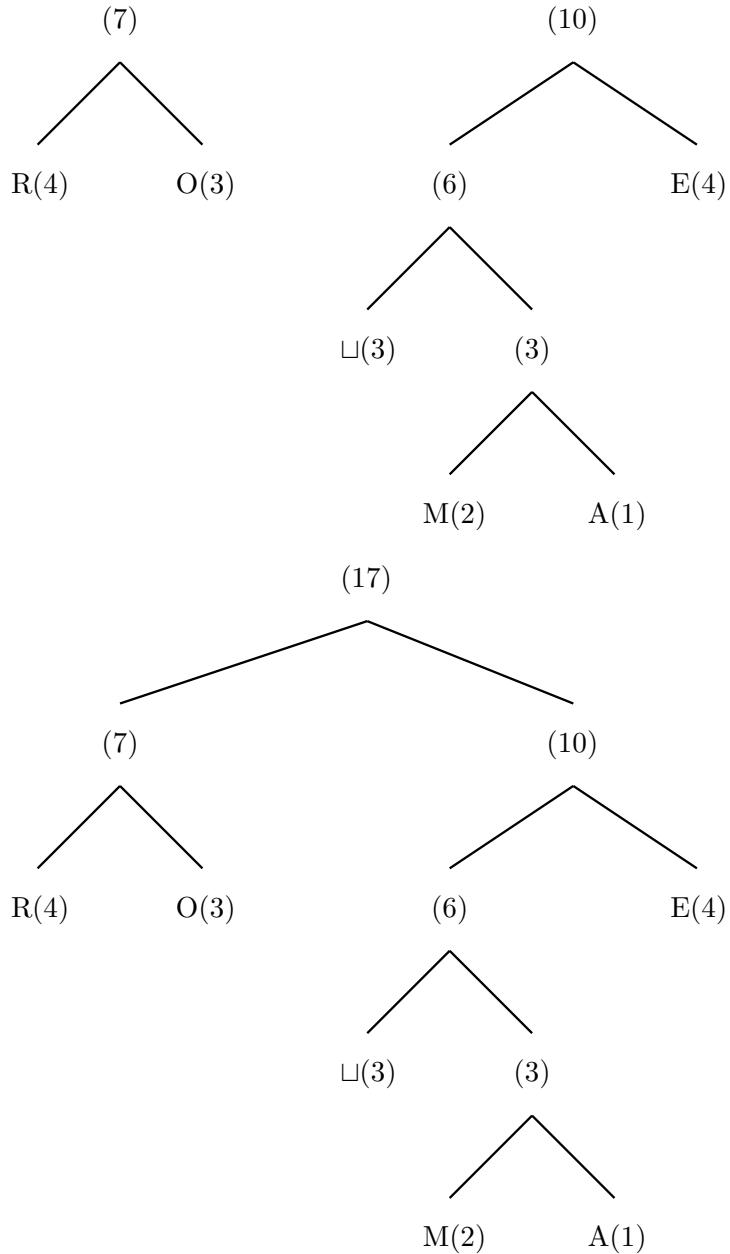


Agora nos nodos com menores frequências temos, um nodo com frequência 3 e dois com frequência 4. Assim vamos juntar o nodo de frequência 3 com um dos nodos de frequência 4. Estes dão origem a um novo nodo de frequência 7.



Neste ponto temos três nodos, um com frequência 7 um com frequência 6 e um com frequência 4. Juntamos assim os nodos de frequência 6 e 4. O nodo resultante desta junção tem frequência 10.

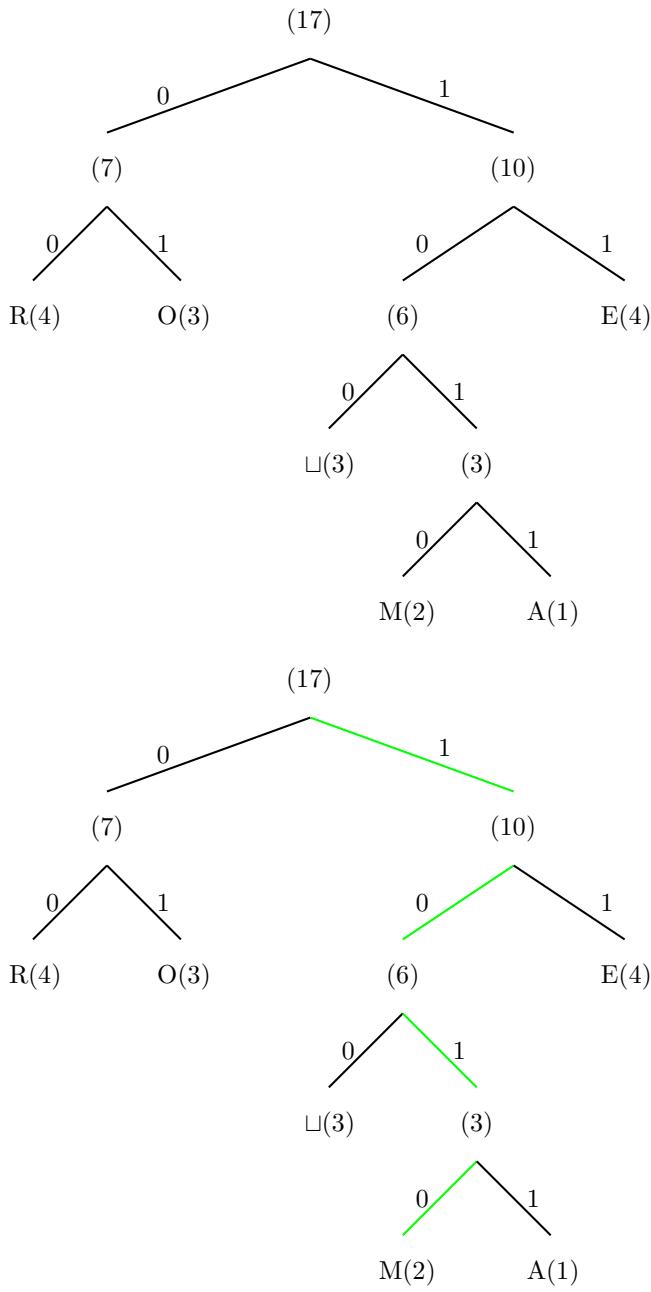
Finalmente restam-nos dois nodos. Juntamos o nodo de frequência 7 com o nodo de frequência 10. Ficamos assim com a raiz da nossa árvore binária com o total de frequências.



Terminamos assim a construção da árvore contendo todos os elementos a codificar. Agora apenas temos de atribuir os valores 0 e 1 aos diferentes ramos da árvore. Neste caso vamos dar o valor 0 aos ramos da esquerda e o valor 1 aos ramos da direita.

Para determinar os códigos de um elemento é só percorrer os caminhos que vão desde a raiz até ao elemento para o qual pretendemos determinar o código, juntando os bits 0 ou 1. Por exemplo, no caso do elemento "M" o código será 1010.

Procedendo de modo semelhante obtemos os restantes códigos.



Símbolo $i$	Código
E	11
Espaço	100
R	00
M	1010
O	01
A	1011

### 6.4.2 Método de Huffman canónico

Para descomprimir um ficheiro com o algoritmo de Huffman temos de conhecer as palavras código de cada símbolo. Estas terão de ser transmitidas no cabeçalho do ficheiro comprimido. Quando os ficheiros utilizam um grande número de símbolos este cabeçalho ocupa uma parte considerável de memória e diminui a performance de compactação. A solução consiste na criação de um código de Huffman canónico.

Canónico aqui denota uma de entre muitas alternativas que pode ser distinguida pois segue regras simples. As regras usadas aqui são as seguintes:

- Códigos mais pequenos têm numericamente (se preenchidos com zeros à direita) valores superiores do que códigos mais longos.
- Dentro do mesmo comprimento, os valores numéricos aumentam com o alfabeto.

Assim, o código mais longo contém apenas zeros, e cada código difere 1 bit do anterior.

---

**Exemplo** Usemos o exemplo do Código de Huffman. Este tinha terminado com a seguinte codificação:

Símbolo $i$	Código
E	11
Espaço	100
R	00
M	1010
O	01
A	1011

Colocando na forma canónica teremos

Símbolo $i$	Código
A	0000
M	0001
Espaço	001
E	01
O	10
R	11

Isto é,

- Código com tamanho 4 começa em 0000;
- Códigos com tamanho 3 começam em  $(0000+2)=0010$ ; Mas como deve ter tamanho 3 remove-se o último 0, ficando 001;

- Códigos com tamanho 2 começam em  $(001+1)=010$ ; Mas como deve ter tamanho 2 remove-se o último 0, ficando 01;
- 

Neste caso de Huffman canónico o cabeçalho pode ser simplificado. Aqui só precisamos de saber o número de bits a usar para codificar cada símbolo.

### 6.4.3 Método de Huffman adaptativo

O método de codificação de Huffman adaptativo evita o cálculo das probabilidades necessários no Huffman standard. Para isso este método usa árvores binárias equilibradas na qual os símbolos vão ser inseridos à medida que ocorrem na fonte a codificar. Visto que a mesma árvore é gerada pelo descodificador não necessitamos de nenhum cabeçalho.

Neste método é usado um símbolo especial que é colocado na árvore logo na altura da inicialização, que é o caractere cujo código vai para o ficheiro a indicar que o caractere está a ocorrer pela primeira vez. A seguir a este código aparece sempre o novo caractere. Com esta informação o descodificador conseguirá gerar a mesma árvore do codificador.

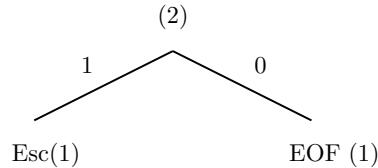
As árvores de Huffman têm de obedecer sempre à seguinte propriedade: Se visitarmos a árvore por nível, da esquerda para a direita e de baixo para cima, as frequências estarão sempre ordenadas de forma não descendente. Assim, símbolos com altas frequências terão códigos mais pequenos. O requerimento de em cada nível as frequências estejam ordenadas da esquerda para a direita serve apenas para simplificar a actualização da árvore.

Os passos a seguir são os seguintes (Vou considerar o ESC o meu código especial):

- Se o caractere é novo:
  - Colocar o código do ESC no ficheiro comprimido seguido do novo caractere;
  - Inserir na árvore sempre o mais abaixo e à esquerda possível (se é novo deverá ficar no local com a menor frequência).
- Se o caractere(X) já existe na árvore.
  - Colocar o código do caractere no ficheiro comprimido;
- Actualizar a árvore:
  - Comparar o peso do caractere X que estamos a codificar (F) com o sucessor na árvore (da esquerda para a direita e de baixo para cima). Se este tiver frequência F+1 ou superior os nodos continuam ordenados e não é necessário alterar nada. Senão, algum sucessor de X tem frequência idêntica a F ou inferior. Neste caso, X deve ser trocado com o último do grupo (excepto X não deve ser trocado com o seu pai).
  - Incrementar a frequência de X de F para F+1. Incrementar as frequências de todos os seus pais.
  - Se X for a raiz pária, senão repetir com o pai do nodo como X.

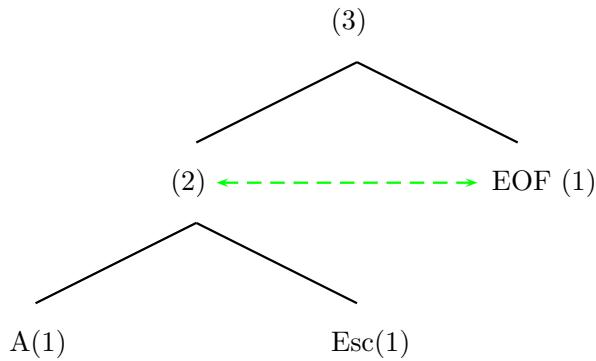
**Exemplo** Vamos codificar a mensagem: "AABBC".

Começamos com a árvore inicializada (a inicialização apresentada aqui é a mesma da implementação do codificador usado nas práticas).

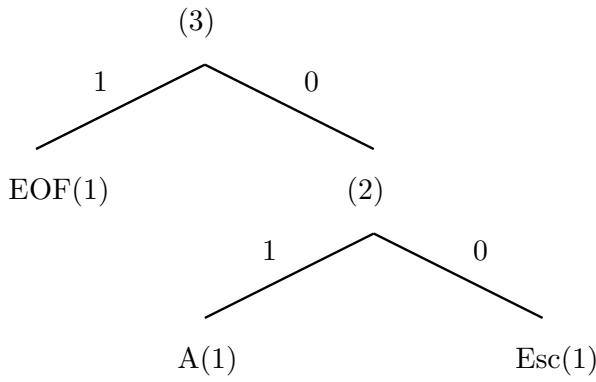


### Iteração 1

- Entra: A
- Ficheiro comprimido: 1A

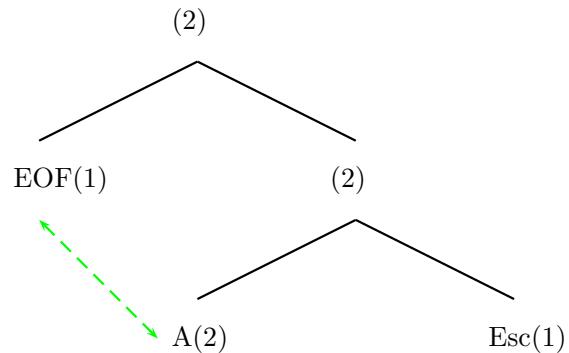


Para equilibrar teremos de trocar o nodo (2) com o nodo do EOF(1)

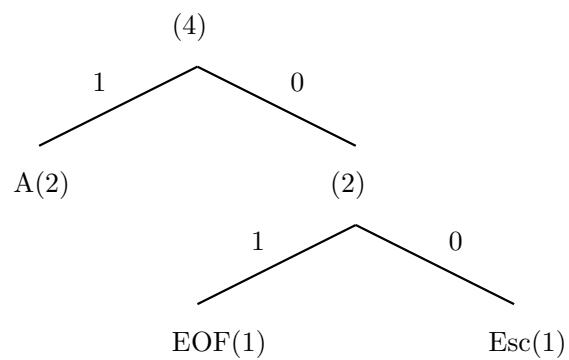


### Iteração 2

- Entra: A
- Ficheiro comprimido: 1A01

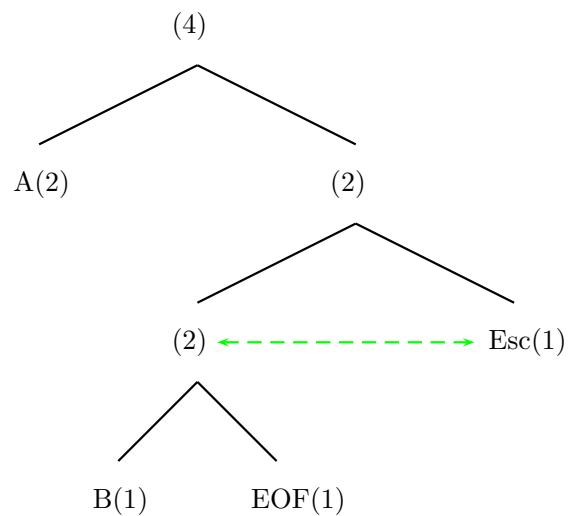


Equilibrando:

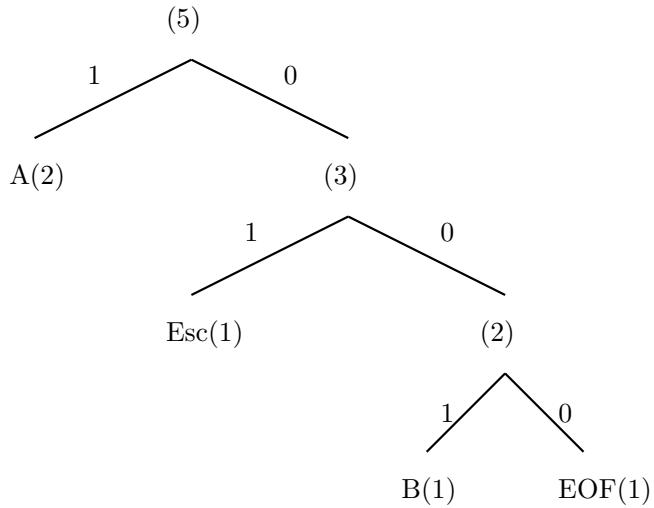


### Iteração 3

- Entra: B
- Ficheiro comprimido: 1A0100B

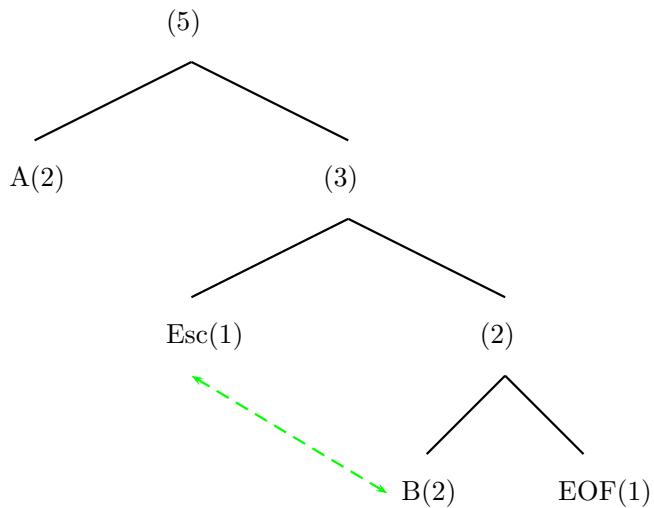


Equilibrando:

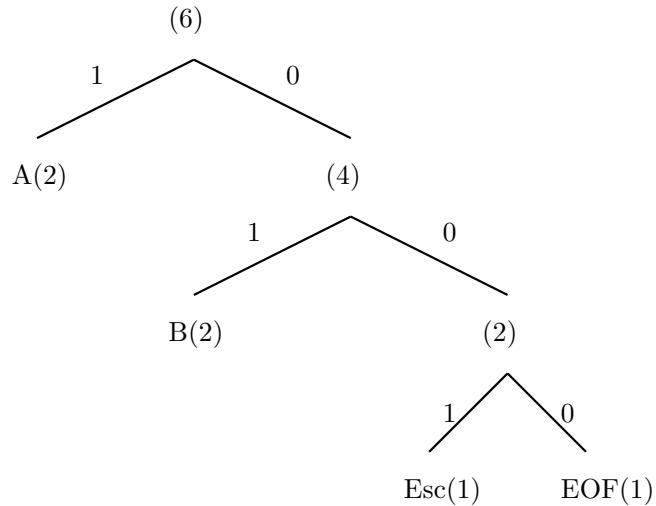


#### Iteração 4

- Entra: B
- Ficheiro comprimido: 1A0100B001

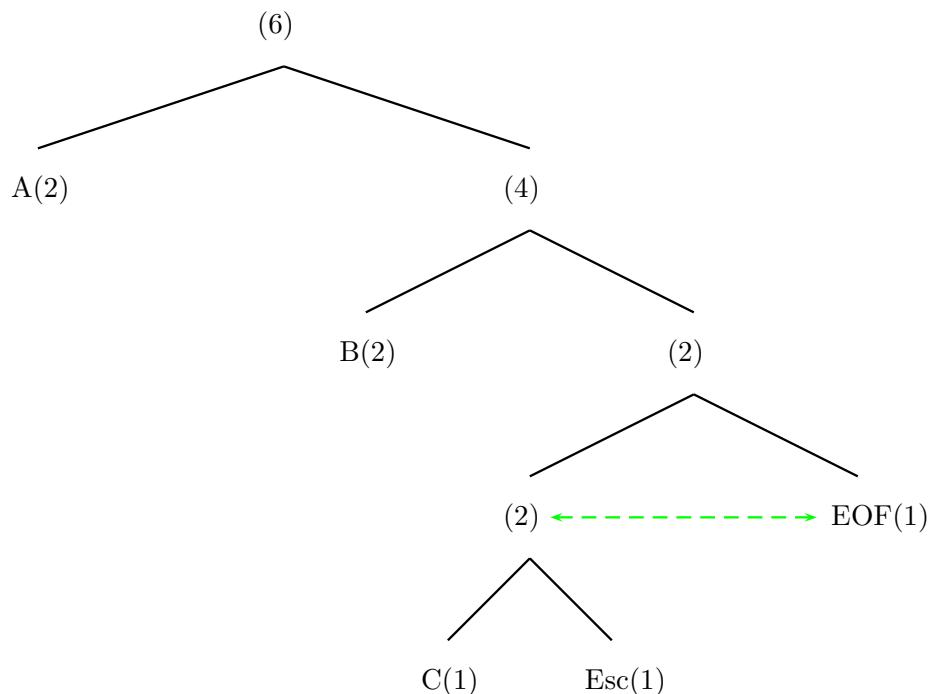


Equilibrando:

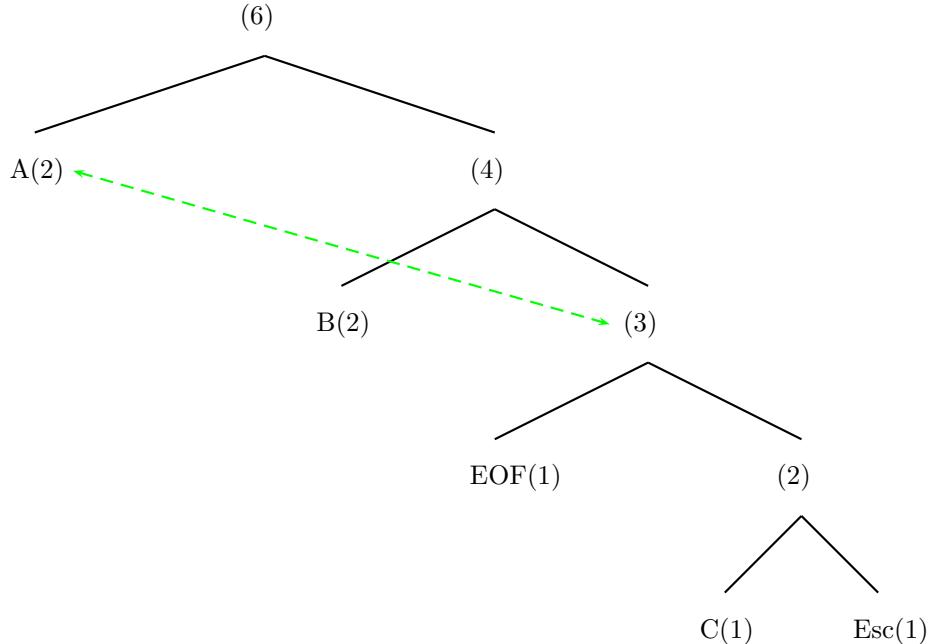


#### Iteração 5

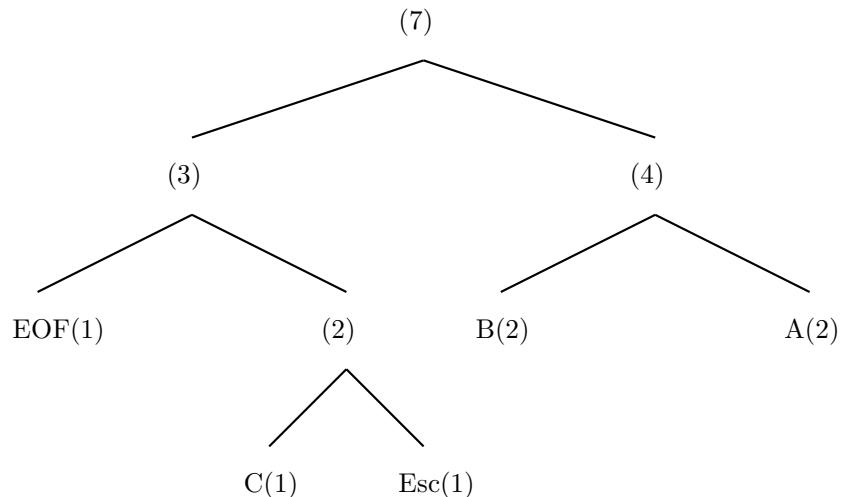
- Entra: C
- Ficheiro comprimido: 1A0100B001001C



Equilibrando:



Equilibrando:



Não havendo mais símbolos a codificar o método termina. O ficheiro comprimido resultante contém **1A0100B001**. Obviamente que o símbolo A e o símbolo B aparecem em binário, no entanto neste exemplo foram sempre deixados os símbolos correspondentes para facilitar a leitura.

---

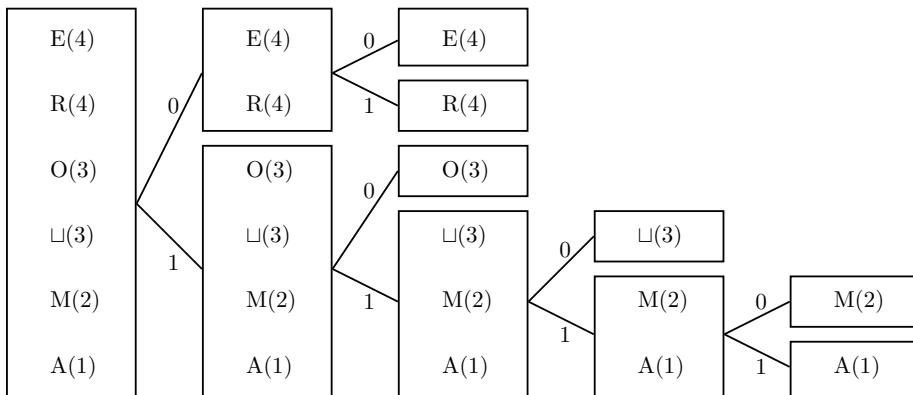
#### 6.4.4 Método de Shannon-Fano

Este é mais um método estatístico. Neste caso é construída uma árvore descendente, partindo sucessivamente a tabela de frequências a partir de valores mais elevados. Para aplicar a um ficheiro o algoritmo de Shannon-Fano percorremos as etapas seguintes:

1. Leitura do ficheiro e cálculo das frequências de cada símbolo;
2. Classificação dos símbolos em função das suas frequências;
3. Divisão das frequências em dois sub-grupos de frequências próximas, de tal forma que a relação da soma das frequências de cada sub-grupo seja o mais próxima possível de 1:  $\sum_1 f_i / \sum_2 f_j \approx 1$ . Esta operação é repetida até que todas as frequências de origem sejam encontradas.
4. Atribuição de um código a cada símbolo (idêntico ao de Huffman);
5. Codificação final.

**Exemplo** Usando o mesmo exemplo usado nos códigos de Huffman teremos:

Símbolo $i$	Frequência $f_i$
E	4
R	4
O	3
Espaço	3
M	2
A	1



Seguindo os caminhos da esquerda para a direita chegamos aos códigos de cada elemento.

Símbolo $i$	Frequência $f_i$	Código
E	4	00
R	4	01
O	3	10
Espaço	3	110
M	2	1110
A	1	1111

### Inconvenientes

Alguns inconvenientes deste método são:

- Leitura do ficheiro para cálculo das frequências;
- Necessidade de guardar as tabelas de códigos;

Uma solução para estes problemas pode passar pela utilização de tabelas estáticas juntando as probabilidades médias representativas para certos tipos de ficheiros. Por exemplo, para texto, a partir de um grande número de amostras podemos calcular a probabilidade de ocorrência das letras de um alfabeto numa língua qualquer.

#### 6.4.5 Codificação Aritmética

A codificação de Huffman é óptima para codificar variáveis com distribuições conhecidas que têm de ser codificadas símbolo a símbolo. No entanto, devido ao facto de os comprimentos dos códigos, no caso dos códigos de Huffman, terem de ser inteiros, pode haver uma perda de até um bit por símbolo, na eficiência do codificador. Podemos aliviar esta perda escolhendo blocos de símbolos de entrada - no entanto, a complexidade desta abordagem aumenta exponencialmente com o comprimento do bloco.

Vamos agora descrever um método de codificação sem esta ineficiência. Na codificação aritmética, em vez de se usar uma sequência de bits para representar um símbolo, representá-lo por um sub intervalo do intervalo unitário.

O código para uma sequência de símbolos é um intervalo cujo comprimento diminui à medida que acrescentamos mais símbolos à sequência. Esta propriedade permite-nos ter um esquema de codificação que é incremental (o código de uma extensão de uma sequência pode ser calculado simplesmente a partir do código da sequência original) e para os quais os comprimentos das palavras de códigos não são restritos a ser inteiros. A motivação da codificação aritmética é baseado na codificação de Shannon-Fano-Elias.

O primeiro passo é calcular, ou pelo menos estimar, a frequência de ocorrências de cada símbolo. Para melhor resultados, as frequências exactas são calculadas lendo o ficheiro de entrada todo no primeiro passo de uma compressão em dois passos. No entanto, se o programa poder obter uma boa estimativa das frequências, o primeiro passo pode ser omitido.

As seguintes regras sumariam os principais passos da codificação aritmética:

1. Começar por definir o intervalo corrente como  $[0, 1)$ .

2. Repetir os passos seguintes para cada símbolo de entrada  $s$ :
    - (a) Dividir o intervalo corrente em sub intervalos cujos comprimentos sejam proporcionais às probabilidades dos símbolos.
    - (b) Selecionar o sub intervalo de  $s$  e defini-lo como o novo intervalo corrente.
  3. Quando todos os símbolos tiverem sido processados desta forma, o output pode ser qualquer número pertencente ao intervalo corrente.
- 

**Exemplo** O primeiro exemplo envolve os três símbolos  $a_1, a_2, a_3$  com probabilidades  $P_1 = 0.4, P_2 = 0.5$  e  $P_3 = 0.1$ , respectivamente. O intervalo  $[0, 1)$  é dividido entre estes três símbolos atribuindo a cada um, um sub intervalo proporcional em tamanho à sua probabilidade. A ordem dos sub intervalos é irrelevante. No nosso exemplo, aos três símbolos são atribuídos os sub intervalos  $[0, 0.4)[0.4, 0.9)[0.9, 1.0)$ .

Para codificar a string  $a_2a_2a_2a_3$ , começamos com o intervalo  $[0, 1)$ . O primeiro símbolo  $a_2$  reduz este sub intervalo no sub intervalo  $[0.4, 0.9)$ .

Subdividindo este intervalo actual nos sub intervalos teremos os sub intervalos  $[0.4, 0.6), [0.6, 0.85)$  e  $[0.85, 0.9)$  ( $0.4 + (0.9 - 0.4) \times 0.4 = 0.6$  e  $0.4 + (0.9 - 0.4) \times 0.9 = 0.85$ ).

O segundo  $a_2$  reduz este sub intervalo da mesma forma, em  $[0.6, 0.85)$ , o terceiro  $a_2$  reduz este em  $[0.7, 0.825)$ , e o  $a_3$  reduz este em  $[0.8125, 0.8250)$ . O código final que o nosso método produz pode ser qualquer valor deste intervalo final.

---

Note que, para cada símbolo processado o intervalo actual torna-se cada vez menor, logo são necessários mais bits para o exprimir, mas o ponto é que o output final é um único número e não consiste em códigos para símbolos individuais. O comprimento médio do código pode ser obtido dividindo o tamanho do output (em bits) pelo tamanho do input (em símbolos). Repare também que as probabilidades usadas no passo 2(a) podem ser alteradas a qualquer altura, visto que podem ser fornecidas por um modelo de probabilidades adaptável.

Vamos de seguida ver mais alguns exemplos.

---

**Exemplo** Vamos ver os passos da compressão da sequência "SWISS MISS". A tabela 6.1 mostra-nos a informação preparada no primeiro passo (o modelo estatístico dos dados). Os cinco símbolos que ocorrem na entrada podem ser organizados por qualquer ordem. Para cada símbolo, contamos as frequências e calculamos as probabilidades. O intervalo  $[0, 1)$  é então dividido entre os símbolos, em qualquer ordem, associando a cada símbolo um sub intervalo de comprimento igual à sua probabilidade.

Por exemplo, o símbolo "S" ficou com o intervalo  $[0.5, 1)$  (de tamanho 0.5), enquanto o sub intervalo de "T" é  $[0.2, 0.4)$  (de tamanho 0.2).

Vamos definir duas variáveis **Low** e **High**, para os limites inferiores e superiores do intervalo actual. O intervalo actual passa então a poder ser definido como **[Low, High]**. Inicialmente **Low** tomará o valor 0 e **High** o valor 1.

Símbolo	Frequência	Probabilidade	Intervalo
S	5	$5/10 = 0.5$	$[0.5, 1.0)$
W	1	$1/10 = 0.1$	$[0.4, 0.5)$
I	2	$2/10 = 0.2$	$[0.2, 0.4)$
M	1	$1/10 = 0.1$	$[0.1, 0.2)$
◻	1	$1/10 = 0.1$	$[0.0, 0.1)$

Tabela 6.1: Frequências e probabilidades dos cinco símbolos.

Vamos também definir **Low(X)** e **High(X)** como os limites inferior e superior do intervalo inicial, associado ao símbolo  $X$ .

Em cada codificação **Low** e **High** serão actualizados da seguinte forma:

$$\begin{aligned} NewHigh &= OldLow + (OldHigh - OldLow) \times High(X) \\ NewLow &= OldLow + (OldHigh - OldLow) \times Low(X) \end{aligned}$$

A tabela 6.2 mostra os passos da codificação da sequência "SWISS MISS". O código final poderá ser qualquer valor do intervalo final. Aqui vamos escolher o valor de **Low**, 0.71753375, do qual apenas os oito dígitos 71753375 precisam de ser escritos no ficheiro de saída.

Símbolo			Calculo de <b>Low(X)</b> e <b>High(X)</b>
S	L	$0.0 + (1.0 - 0.0) \times 0.5 = 0.5$	
	H	$0.0 + (1.0 - 0.0) \times 1.0 = 1.0$	
W	L	$0.0 + (1.0 - 0.5) \times 0.4 = 0.70$	
	H	$0.0 + (1.0 - 0.5) \times 0.5 = 0.75$	
I	L	$0.7 + (0.75 - 0.70) \times 0.2 = 0.71$	
	H	$0.7 + (0.75 - 0.70) \times 0.4 = 0.72$	
S	L	$0.71 + (0.72 - 0.71) \times 0.5 = 0.715$	
	H	$0.71 + (0.72 - 0.71) \times 1.0 = 0.72$	
S	L	$0.715 + (0.72 - 0.715) \times 0.5 = 0.7175$	
	H	$0.715 + (0.72 - 0.715) \times 1.0 = 0.720$	
◻	L	$0.7175 + (0.72 - 0.7175) \times 0.0 = 0.7175$	
	H	$0.7175 + (0.72 - 0.7175) \times 0.1 = 0.71775$	
I	L	$0.717525 + (0.71755 - 0.717525) \times 0.2 = 0.717530$	
	H	$0.717525 + (0.71755 - 0.717525) \times 0.4 = 0.717535$	
S	L	$0.7175325 + (0.717535 - 0.717530) \times 0.5 = 0.7175320$	
	H	$0.7175325 + (0.717535 - 0.717530) \times 1.0 = 0.717535$	
S	L	$0.7175325 + (0.717535 - 0.7175325) \times 0.5 = 0.71753375$	
	H	$0.7175325 + (0.717535 - 0.7175325) \times 1.0 = 0.717535$	

Tabela 6.2: Processo de codificação aritmética.

O descodificador começa também por construir a tabela 6.1. Depois basta-lhe verificar em que subintervalo o código se encontra à medida que reconstrói a tabela 6.2.

**Nota** O descodificar não sabe quando terminar de descodificar. Para isso existem várias soluções. Podemos por exemplo usar um símbolo de fim de codificação, desta forma o descodificador quando descodifica esse símbolo sabe que pode terminar o processo de descodificação. Podemos também simplesmente comunicar ao descodificador a quantidade de símbolos que ele deverá decodificar.

O comprimento do intervalo final é o produto das probabilidades de todos símbolos no texto.

Na codificação aritmética adaptativa, podemos assumir probabilidades iguais para todos símbolos no início da codificação/descodificação, e fazer a contagem dos símbolos para nos aproximarmos das suas probabilidades reais.

## 6.5 Algoritmos do tipo dicionário

Os métodos de compressão baseados em dicionários são caracterizados principalmente por :

- Não necessitarem de conhecer a estatística dos dados a comprimir.
- Não usarem códigos de comprimento variável.
- Utilizarem sequências de símbolos de comprimento variável.
- Muitos dos programas de compressão de dados mais conhecidos baseiam-se nestas técnicas: pkzip, zoo, arj, compress, gzip, ...

A sua invenção deveu-se a Jacob Ziv e Abraham Lempel (J. Ziv and A. Lempel, A universal algorithm for sequential data compression, IEEE Trans. on Information Theory, 1977, 23, pp. 337-343), em finais dos anos 70: LZ77 e LZ78. Desde então, muitas variantes foram propostas: LZH, LZSS,LZW (Lempel, Ziv, Welch). Este último método deveu-se a Terry Welch que em 1984 publicou um artigo onde alguns dos problemas associados à compressão LZ78 foram eliminados.

Este é o algoritmo de compressão sem perdas mais utilizado. Pode ser usado para comprimir texto, código binário, código executável, imagens (GIF, TIFF, PostScript).

### 6.5.1 Algoritmo de codificação LZW

Inicialmente, os símbolos do alfabeto são inseridos no dicionário. O algoritmo é o seguinte:

1. Inicializa uma sequência (S) a vazio;
2. Lê símbolo (x) da mensagem a codificar;
3. Se símbolo é o EOF vai para ponto 7.
4. Gera sequência (Sx) da concatenação de S com x ( $Sx = S+x$ );
5. Se  $Sx$  puder ser encontrada no dicionário:

- (a)  $S = SX$ ;  
 (b) vai para ponto 2.
6. Se  $SX$  não puder ser encontrada no dicionário:  
 (a) é emitido o código de  $S$ ;  
 (b)  $SX$  é colocada no dicionário;  
 (c) e  $S$  é inicializada com o símbolo  $x$ .  
 (d) Vai para ponto 2;
7. é emitido o código de  $S$ ;
- 

**Exemplo** Codificação da mensagem "aaabaaaadaabaado".

Inicialmente, o dicionário contém todos os símbolos do alfabeto.

Dicionário inicial							
0	...	97	98	99	100	...	255
nul	...	a	b	c	d	...	

Vejamos as várias iterações da codificação:

**Iteração 1:** Mensagem: **a** aabaaaadaabaado

- $S = ;$
- $x = 'a';$
- $SX = S+x="a"$
- $S = SX = "a";$

Dicionário no final da Iteração 1							
0	...	97	98	99	100	...	255
nul	...	a	b	c	d	...	

**Iteração 2:** Mensagem: **aa** abaaaadaabaado

- $x = 'a';$
- $SX = S+x="aa"$
- Emito  $S: 97$
- Junto ao Dicionário: "aa"
- $S = x = "a";$

Dicionário no final da Iteração 2							
0	...	97	98	99	100	...	256
nul	...	a	b	c	d	...	aa

**Iteração 3:** Mensagem: **a aab** aaadaabaado

- $x = 'a'$ ;
- $Sx = S + x = "aa"$
- $S = Sx = "aa"$
- $x = 'b'$ ;
- $Sx = S + x = "aab"$
- Emito  $S: 256$
- Junto ao Dicionário: "aab"
- $S = x = "b"$ ;

Dicionário no final da Iteração 3							
...	97	98	99	100	...	256	257
...	a	b	c	d	...	aa	aab

**Iteração 4:** Mensagem: aaa **ba** aadaabaado

- $x = 'a'$ ;
- $Sx = S + c = "ba"$
- Emito S: 98
- Junto ao Dicionário: "ba"
- $S = x = "a"$ ;

Dicionário no final da Iteração 4									
...	97	98	99	100	...	256	257	258	
...	a	b	c	d	...	aa	aab	ba	

**Iteração 5:** Mensagem: aaab **aaa** daabaado

- $x = 'a'$ ;
- $Sx = S + x = "aa"$
- $S = Sx = "aa"$ ;
- $x = 'a'$ ;
- $Sx = S + x = "aaa"$ ;
- Emito S: 256
- Junto ao Dicionário: "aaa"
- $S = x = "a"$ ;

Dicionário no final da Iteração 5									
...	97	98	99	100	...	256	257	258	259
...	a	b	c	d	...	aa	aab	ba	aaa

**Iteração 6:** Mensagem: aaabaa **ad** aabaado

- $x = 'd'$ ;
- $Sx = S + x = "ad"$
- Emito S: 97
- Junto ao Dicionário: "ad"
- $S = x = "d"$ ;

Dicionário no final da Iteração 6										
...	97	98	99	100	...	256	257	258	259	260
...	a	b	c	d	...	aa	aab	ba	aaa	ad

**Iteração 7:** Mensagem: aaabaaa **da** aabaado

- $x = 'a'$ ;
- $Sx = "da"$
- Emito S: 100
- Junto ao Dicionário: "da"
- $S = "a"$ ;

Dicionário no final da Iteração 7											
...	97	98	99	100	...	256	257	258	259	260	261
...	a	b	c	d	...	aa	aab	ba	aaa	ad	da

**Iteração 8:** Mensagem: aaabaaaad **aaba** ado

- $x = 'a'$ ;
- $Sx = "aa"$ ;
- $S = "aa"$ ;
- $x = 'b'$ ;
- $SX = "aab"$ ;
- $S = "aab"$ ;
- $x = 'a'$ ;
- $Sx = "aaba"$ ;
- Emito S: 257
- Junto ao Dicionário: "aaba"
- $S = "a"$ ;

Dicionário no final da Iteração 8							
...	256	257	258	259	260	261	262
...	aa	aab	ba	aaa	ad	da	aaba

**Iteração 9:** Mensagem: aaabaaaadaab **aad** o

- $x = 'a'$ ;
- $Sx = "aa"$ ;
- $S = "aa"$ ;
- $x = 'd'$ ;
- $SX = "aad"$ ;
- Emito S: 256
- Junto ao Dicionário: "aad"
- $S = "d"$ ;

Dicionário no final da Iteração 9								
...	256	257	258	259	260	261	262	263
...	aa	aab	ba	aaa	ad	da	aaba	aad

**Iteração 10:** Mensagem: aaabaaaadaabaa **do**

- $x = 'o'$ ;
- $Sx = "do"$ ;
- Emito S: 100
- Junto ao Dicionário: "do"
- $S = "o"$ ;
- $x = 'EOF'$ ;
- Emito S: 111

Dicionário no final da Iteração 10									
...	256	257	258	259	260	261	262	263	264
...	aa	aab	ba	aaa	ad	da	aaba	aad	do

No final o ficheiro comprimido contém os códigos: 97 256 98 256 97 100 257 256 111.

O tamanho do ficheiro será: número de códigos  $\times \log_2 TamanhoDicionario$ ; neste caso  $9 \times \log_2 TamanhoDicionario$ .

---

### 6.5.2 Algoritmo de descodificação LZW

Inicialmente, os símbolos do alfabeto são inseridos no dicionário. A operação do algoritmo é a seguinte:

1. Inicializa um contador COUNTD = 256;
  2. Lê código (CARCODE);
  3.  $D[\text{COUNTD}] = D[\text{CARCODE}]$ ;
  4. escreve no ficheiro  $D[\text{CARCODE}]$ ;
  5. Lê código (CARCODE); Se EOF termina, senão continua;
  6.  $D[\text{COUNTD}] = D[\text{COUNTD}] + D[\text{CARCODE}][0]$ ;
  7. COUNTD++;
  8. Volta ao ponto 3
- 

**Exemplo** Descodificação de: 97 256 98 256 97 100 257 256 100 111.

Dicionário inicial							
0	...	97	98	99	100	...	255
nul	...	a	b	c	d	...	

**Iteração 1:**

- Código Recebido: 97
- Mensagem: a?

Dicionário no final da Iteração 1							
0	...	97	98	99	100	...	256
nul	...	a	b	c	d	...	a?

**Iteração 2:**

- Código Recebido: 256
- Mensagem: a aa?

Dicionário no final da Iteração 2							
...	97	98	99	100	...	256	257
...	a	b	c	d	...	aa	aa?

**Iteração 3:**

- Código Recebido: 98
- Mensagem: aaa b?

Dicionário no final da Iteração 3								
...	97	98	99	100	...	256	257	258
...	a	b	c	d	...	aa	aab	b?

**Iteração 4:**

- Código Recebido: 256
- Mensagem: aaab aa?

Dicionário no final da Iteração 4									
...	97	98	99	100	...	256	257	258	259
...	a	b	c	d	...	aa	aab	ba	aa?

**Iteração 5:**

- Código Recebido: 97
- Mensagem: aaabaa a?

Dicionário no final da Iteração 5										
...	97	98	99	100	...	256	257	258	259	260
...	a	b	c	d	...	aa	aab	ba	aaa	a?

**Iteração 6:**

- Código Recebido: 100
- Mensagem: aaabaaa d?

Dicionário no final da Iteração 6											
...	97	98	99	100	...	256	257	258	259	260	261
...	a	b	c	d	...	aa	aab	ba	aaa	ad	d?

**Iteração 7:**

- Código Recebido: 257
- Mensagem: aaabaaaad aab?

Dicionário no final da Iteração 7							
...	256	257	258	259	260	261	262
...	aa	aab	ba	aaa	ad	da	aab?

**Iteração 8:**

- Código Recebido: 256
- Mensagem: aaabaaaadaaab aa?

Dicionário no final da Iteração 8								
...	256	257	258	259	260	261	262	263
...	aa	aab	ba	aaa	ad	da	aaba	aa?

**Iteração 9:**

- Código Recebido: 100
- Mensagem: aaabaaaadaaabaa d?

Dicionário no final da Iteração 9									
...	256	257	258	259	260	261	262	263	264
...	aa	aab	ba	aaa	ad	da	aaba	aad	d?

**Iteração 10:**

- Código Recebido: 111
- Mensagem: aaabaaaadaabaado

Dicionário no final da Iteração 10									
...	256	257	258	259	260	261	262	263	264
....	aa	aab	ba	aaa	ad	da	aaba	aad	do

Assim, terminamos a descodificação da mensagem "aabaaaadaabaado".

---

### Vantagens do LZW

Este algoritmo apresenta duas vantagens decisivas:

- A criação do dicionário e a compressão do ficheiro são realizadas à medida que o ficheiro é lido, sem necessitar de uma leitura prévia (assim o tempo de compressão é reduzido).
- O dicionário não é adicionado ao ficheiro comprimido (maior taxa de compressão).

## 6.6 Os métodos mistos: estatísticos e de dicionário

Quando dois métodos de compressão eficazes têm recurso a procedimentos completamente autónomos, é evidentemente tentador combinar as vantagens para acumular a eficácia. Foi o que foi realizado com os métodos de compressão estatísticos e de dicionário. Num primeiro passo, aplicamos aos dados de origem um algoritmo do tipo dicionário, que explora, como acabamos de verificar, as sequências repetitivas. Num segundo tempo, o resultado obtido é submetido a um algoritmo de compressão estatístico que permite obter uma segunda redução de volume do ficheiro.

Esta técnica mista permite obter bons resultados (50 a 65 % sobre a maioria dos ficheiros) para, relembrarmos uma compressão sem perdas. Ela é utilizada por muitas das ferramentas de compressão do domínio público tais como, ARJ, LHA, PKZIP, UC, GZIP, WINZIP, ou ZOO.

## 6.7 Conclusões

Por vezes o tamanho dos ficheiros é tal que temos de aceitar perder alguma informação menos importante no que diz respeito à percepção visual e auditiva humana. Temos então a compressão com perdas que é capaz de reduzir o tamanho dos ficheiros em proporções muito superiores aquelas conseguidas com os métodos de compressão tradicionais, preservando uma boa ou muito boa qualidade de reprodução.

Notemos que estas práticas de compressão com perdas, não excluem a utilização complementar destes métodos de compressão sem perdas. O formato GIF por exemplo utiliza uma compressão LZW.

Os algoritmos de compressão com perdas serão analisados nos capítulos consagrados à imagem, ao som e ao vídeo.

## 6.8 Problemas

1. Considere a variável aleatória  $X$  que representa "a cor de uma bola retirada de uma urna". Considere as seguintes possibilidades.
  - (a) Uma urna com 10 bolas pretas e 10 bolas brancas.
  - (b) Uma urna com 5 bolas pretas, 5 bolas brancas, 5 bolas vermelhas.
  - (c) Uma urna com 1 bola preta e 9 brancas.
  - (d) Uma urna com 10 bolas brancas.

Indique qual das seguintes afirmações é verdadeira:

- Em (a)  $H(X) < 0.5$  e em (b)  $H(X) \geq 0.5$ .
  - Em (d)  $H(X) = 0$  e em (b)  $H(X) \geq 0.5$ .
  - $H(X)$  em (a) é inferior a  $H(X)$  em (c).
  - A menor entropia é no caso (c).
  - Nenhuma das anteriores.
2. (1.0 valores) Considere o par de variáveis aleatórias (v.a.)  $X$  e  $Y$ , binárias (isto é, com valores em  $\{0, 1\}$ ), cuja função de probabilidade conjunta é dada por:  $p(X = 0, Y = 0) = 1/4$ ,  $p(X = 0, Y = 1) = 1/4$ ,  $p(X = 1, Y = 0) = 1/2$ . Qual das seguintes afirmações é verdadeira?
    - $H(Y) = 1$ .
    - $H(X) = 1$  e  $H(Y|X = 1) = 0$ .
    - $H(X) = 1$  e  $H(Y|X) = 0$ .
    - $H(X, Y) = 2$ .
    - $H(Y, X) = 2$ .
    - Nenhuma das afirmações está correcta.
  3. (1.0 valores) Uma fonte gera símbolos do alfabeto  $\{A, B, C, D, E\}$ , com probabilidades  $\{0.5, 0.2, 0.15, 0.1, 0.05\}$ . Qual dos seguintes códigos binários é óptimo para esta fonte?
    - $\{11, 01, 000, 001, 101\}$
    - $\{00, 01, 11, 100, 101\}$
    - $\{0, 10, 110, 1110, 1111\}$
    - $\{0, 01, 110, 1110, 1111\}$
    - Nenhuma das anteriores.
  4. (1.0 valores) Apenas um dos códigos seguintes pode ser um código de Huffman. Qual?
    - $\{0, 10, 11\}$
    - $\{00, 01, 10, 110\}$
    - $\{01, 10\}$
    - $\{0, 10, 01, 11, 110, 111\}$

5. (1.0 valores) Uma fonte gera símbolos do alfabeto  $\{A, B, C, D, E\}$ , com probabilidades  $\{0.3, 0.2, 0.2, 0.15, 0.15\}$ . Qual dos seguintes códigos binários é óptimo para esta fonte?
- $\{1, 01, 001, 0000, 0001\}$
  - $\{00, 01, 11, 100, 101\}$
  - $\{0, 10, 100, 1110, 1111\}$
  - $\{0, 01, 110, 1110, 1111\}$
  - Nenhuma das anteriores.
6. (1.0 valores) Apenas um dos códigos seguintes pode ser um código de Huffman. Qual?
- $\{00, 10, 11\}$
  - $\{00, 01, 10, 110\}$
  - $\{0, 10, 110, 1110, 11110, 111110, 1111110, 11111110, 11111111\}$
  - $\{10, 01, 11, 000, 00100, 00101, 00110, 001110\}$
7. Considere duas fontes com alfabeto  $\{A, B, C, D\}$ . A primeira com probabilidades  $p(A) = 0.25, p(B) = 0.25, p(C) = 0.25, p(D) = 0.25$ , e a segunda com probabilidades  $p(A) = 0.5, p(B) = 0.3, p(C) = 0.1, p(D) = 0.1$ .
- (a) Desenhe um código de Huffman binário para cada uma destas fontes.
  - (b) Qual dos códigos é mais eficientes; Explique usando a largura média e a entropia.
  - (c) Calcule e explique as taxas de compressão obtidas.
  - (d) Supondo que a fonte continha um total de  $256 \times 256$  símbolos qual o tamanho aproximado, em bits, dessa fonte em cada um dos casos.
8. Considere que temos a codificar uma fonte com alfabeto  $\{A, B, C, D, E, F\}$  com probabilidades  $p(A) = 0.3, p(B) = 0.2, p(C) = 0.15, p(D) = 0.15, p(E) = 0.1$  e  $p(F) = 0.1$ .
- (a) Desenhe um código de Huffman binário para esta fonte e calcule o seu comprimento médio. Compare com a entropia.
  - (b) Supondo que a fonte continha um total de  $256 \times 256$  símbolos qual o tamanho aproximado, em bits, dessa fonte. Compare com o tamanho da fonte no caso de se usarem 8 bits por símbolo. Qual a taxa de compressão conseguida com o código de Huffman?
9. Considere os códigos  $C1 = \{00, 01, 10, 11\}$  e  $C2 = \{0, 01, 110, 111\}$ .
- (a) Indique qual destes códigos não é unicamente descodificável e explique porquê.
  - (b) Indique justificando se algum destes códigos poderá ser considerado código de Huffman.
10. Diga justificando se a afirmação seguinte é verdadeira:
- A entropia de uma fonte que usa 8 símbolos diferentes é sempre superior a 3 ( $H > 3$ ).*

11. Considere a seguinte mensagem:

*AABCDEAABCDEAABCDEAABCDEAABCDAABA*

- (a) Codifique esta mensagem usando codificação de Huffman.
- (b) Codifique esta mensagem usando codificação de Shannon-Fano.
- (c) Qual dos códigos é mais eficiente? Já esperava esse resultado?
- (d) Calcule as taxas de compressão obtidas com os dois códigos calculados anteriormente.
- (e) Use a codificação LZW para codificar a mensagem acima, sabendo que o dicionário inicialmente contém apenas dois valores, *A* com código 0000 e *B* com código 0001, *C* com o código 0010, *D* com o código 0011 e *E* com o código 0100. Apresente a mensagem codificada com o seu código considerando que cada código usa quatro bits (logo o dicionário poderá conter no máximo 16 símbolos diferentes).

12. Considere a mensagem *AAABAABBBBAA*.

- (a) Use a codificação LZW para codificar a mensagem acima, sabendo que o dicionário inicialmente contém apenas dois valores, *A* com código 0 e *B* com código 1.
- (b) Supondo que vamos considerar um dicionário de tamanho máximo 16 indique quantos bits ocupa a mensagem codificada com o seu código LZW.

13. Considere:

- (a) A codificação da mensagem "MISS MISS" usando o codificador aritmético.
- (b) A codificação da mensagem "MISS MISS" usando a codificação de Huffman.
- (c) A descodificação da mensagem 001 000 010 101 110 que foi codificada usando o codificador LZW com um dicionário de tamanho 8 com os caracteres I, M, S no dicionário inicial.

Indique qual das seguintes afirmações é verdadeira:

- A codificação aritmética dá um valor inferior a 0.7175
  - A codificação mais eficiente da mensagem "MISS MISS" é a do ponto (b).
  - A mensagem descodificada no ponto (c) é "MISS MISS".
  - Todas as afirmações anteriores estão correctas.
  - Nenhuma das anteriores.
14. (1.0 valores) Considere a utilização do algoritmo LZW com um dicionário de tamanho 16 e com apenas os caracteres A e B no dicionário inicial. A descodificação de "0000001000110000" é:
- AABBA

- AABBBA
  - AAAAAAA
  - AAAAABA
  - Nenhuma das anteriores.
15. (1.0 valores) Considere a utilização de um codificador aritmético para descodificar a seguinte sequência: "000110". Considere que os símbolos foram ordenados por ordem decrescente de probabilidade e dentro desta por ordem alfabética. Após a codificação do terceiro 0 o estado do intervalo será:
- [0, 0.296296]
  - [0.197530, 0.296296]
  - [0, 0.444444]
  - [0.131687, 0.296296]
  - Nenhuma das anteriores.
16. (1.0 valores) Considere a utilização do algoritmo LZW com um dicionário de tamanho 16 e com apenas os caracteres A e B no dicionário inicial. A descodificação de "AAAAAAA" é:
- 0000001100110000
  - 0000001001110000
  - 0001001000110000
  - 0000001000110000
  - Nenhuma das anteriores.
17. (1.0 valores) Considere a utilização de um codificador aritmético para descodificar a seguinte sequência: "00011100". Considere que os símbolos foram ordenados por ordem decrescente de probabilidade e dentro desta por ordem alfabética. Após a codificação do segundo 0 o estado do intervalo será:
- [0, 0.39062]
  - [0.244140, 0.335693]
  - [0, 0.244140]
  - [0, 0.335693]
  - Nenhuma das anteriores.



## Capítulo 7

# Representação de Imagem Digital

Na compressão com perdas os dados depois de descomprimidos não são os mesmos que os originais, mas são uma aproximação destes. Isto permite-nos níveis de compressão muito mais elevados do que a compressão sem perdas.

Podemos ver na figura 7.1 um esquema onde depois do processo de compressão se obtém uma aproximação  $\hat{x}$  da informação original  $x$ .

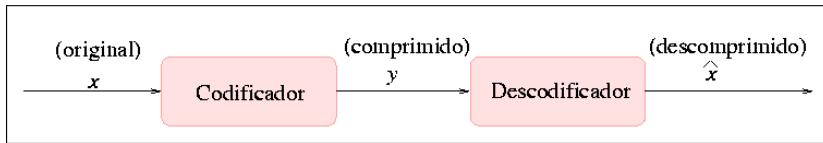


Figura 7.1: Esquema de compressão descompressão.

### 7.1 Medidas de distorção

Visto que na compressão com perdas obtenho uma aproximação da informação original, neste caso da imagem original, esta não é exactamente igual à imagem original. Para sabermos a qualidade da imagem obtida usam-se medidas de distorção. Apresentamos de seguida algumas dessas medidas.

**MSE** Erro médio quadrado (Mean square Error):  $\sigma^2$ , com,

$$\sigma^2 = \frac{1}{N} \sum_{n=1}^N (x_n - \hat{x}_n)^2 \quad (7.1)$$

onde  $x_n$  é a sequência de dados de entrada,  $\hat{x}_n$  é a sequência de dados reconstruídos e  $N$  é o número de dados.

**SNR** Signal to Noise Ratio:

$$SNR = 10 \log_{10} \frac{\sigma_x^2}{MSE} dB \quad (7.2)$$

onde  $\sigma_x^2$  é a variância da sequência original.

**PSNR** Peak Signal to Noise Ratio,

$$PSNR = 10 \log_{10} \frac{m^2}{MSE} \quad (7.3)$$

sendo  $m$  o maior valor possível para um pixel.

O PSNR é medido em dB (tal como o SNR).

---

**Exemplo** Normalmente em imagens codificadas JPEG: diferenças de 0.5 a 1 dB são perceptíveis; Imagens com qualidade aceitável começam nos 25-30 dB; A 35-40 dB não há praticamente diferenças visíveis.

---

Podemos ainda calcular a taxa de compressão que nos permite comparar o número de bits com e sem compressão. Esta é calculada como apresentado na equação 7.4.

$$\text{Taxa de compressão} = \frac{N_1 N_2 B}{\|c\|} \quad (7.4)$$

$N_1$  e  $N_2$  representam as dimensões da imagem e  $B$  o número de bits usados para representar cada pixel. O  $\|c\|$  representa o tamanho da trama binária comprimida  $c$ .

Para compressão com perdas, o débito binário tem mais significado do que taxa de compressão. Este calcula-se segundo a equação 7.5

$$\text{Débito binário} = \frac{\|c\|}{N_1 N_2} \quad (7.5)$$


---

**Exemplo** Em imagens codificadas em JPEG normalmente imagens codificadas acima dos 1.5 bpp não se conseguem distinguir das originais, imagens codificadas com 1bpp possuem uma qualidade alta, imagens codificadas a 0.5 bpp possuem uma qualidade média e imagens codificadas com 0.25 bpp possuem uma qualidade utilizável.

---

## 7.2 Sistema de compressão

A figura 7.2 representa uma estrutura típica de um sistema de compressão:

Nesta figura podemos ver que existem três blocos principais no codificador e três blocos que realizam as operações inversas no descodificador.

**Transformada** Este passo tem por objectivo representar os dados da imagem de forma a eliminar as redundâncias estatísticas. Neste passo os valores de entrada são transformados em coeficientes. Muitos destes coeficientes são zero. Assim, a energia está compactada em poucos coeficientes.

A transformada é normalmente invertível.

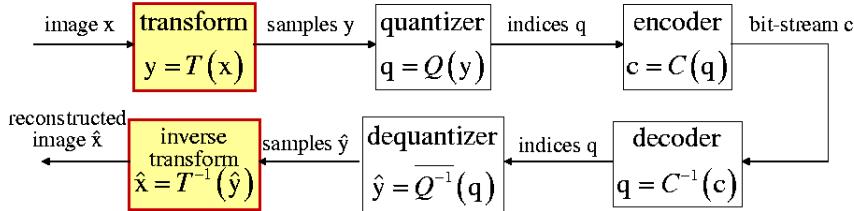


Figura 7.2: Sistema de compressão.

**Quantização** Este passo tem por objectivo reduzir o número de valores de amplitude possíveis para codificação.

A quantização não é invertível, introduzindo erro de quantização.

**Codificação** Este passo tem por objectivo explorar a não uniformidade da distribuição de probabilidade dos índices de quantização.

Este passo não implica perdas.

Iremos estudar mais detalhadamente cada um destes blocos, mas antes notemos que no caso das imagens com cor é normalmente realizado antes do primeiro bloco que diz respeito à Transformada uma transformação de cor. Começamos por falar desta e depois entraremos nos blocos que apresenta a figura 7.2.

### 7.2.1 Transformada de cor

Podemos obter uma redução considerável do débito binário se em vez de codificarmos os valores RGB codificarmos separadamente os diferentes componentes do sinal (luminância e crominância). Na codificação por componentes transforma-se a imagem de cor para um modelo que separe a luminância de crominância. Temos assim um plano Y para a luminância e dois planos para a crominância.

Sabendo que o olho é mais sensível à luminância do que à crominância podemos reduzir as matrizes de crominância: para isso calculamos a média de blocos quadrados de 4 pixels. Esta redução assegura já uma taxa de compressão de 2. Para uma imagem de tamanho  $640 \times 480$  as matrizes de crominância serão de apenas  $320 \times 240$ .

### 7.2.2 Transformadas

Uma imagem é representada como uma matriz bi-dimensional de coeficientes, cada coeficiente representa o nível do brilho nesse ponto. Ao olhar de um perspectiva mais elevada, não podemos diferenciar os coeficientes como mais importantes, e menos ou pouco importantes. Mas pensando mais intuitivamente, isto é possível. A maioria das imagens naturais têm variações suaves de cor, com os detalhes mais finos a serem representados como variações bruscas "sharp edges" entre as variações suaves. Tecnicamente, as variações suaves na cor podem ser denominadas como variações de baixa frequência e as variações bruscas como variações de alta frequência.

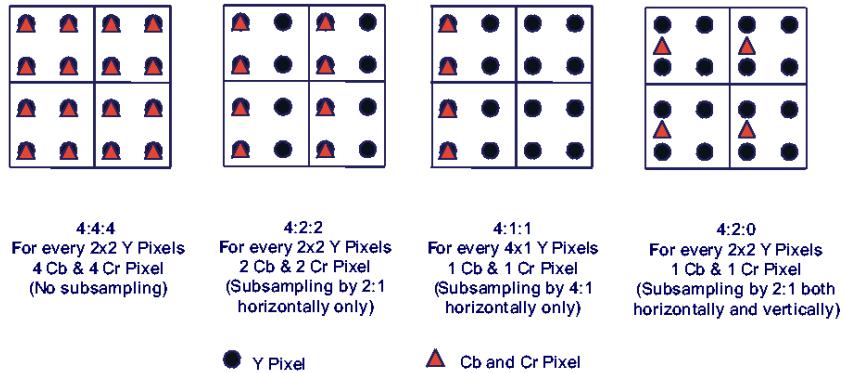


Figura 7.3: YCbCr em 4:4:4, em 4:2:2 em 4:1:1 e em 4:2:0.

Os componentes de baixa frequência (variações suaves) constituem a base de uma imagem, e as altas frequências (os "edges" que fornecem o detalhe) juntam detalhe para refinar a imagem, desse modo fornecendo uma imagem detalhada. Consequentemente, as variações suaves exigem mais importância que os detalhes.

Separar as variações suaves dos detalhes pode ser efectuado de várias formas.

A codificação por transformadas constitui uma componente integral das aplicações de processamento de imagem/vídeo. A codificação por transformada assenta na permaisa que pixeis numa imagem exibem um certo grau nível de correlação com os pixeis vizinhos. De forma similar, num sistema de transmissão de vídeo, pixeis adjacentes em frames consecutivos mostram grande nível de correlação. Consequentemente, estas correlações podem ser exploradas para prevêr o valor de um pixel a partir dos respectivos vizinhos. Uma transformação é, desta forma, definida para mapear estes dados espaciais (correlacionados) em coeficientes transformados (não correlacionados).

Os codificadores fonte exploram esta redundância para obter melhor compressão. As transformadas descorrelacionam os dados das imagens reduzindo (e nalguns casos eliminando) a redundância interpixel (espacial, geométrica e inter-frame).

Rpare nas imagens seguintes:



Figura 7.4: Imagens originais.

Estas duas imagens possuem os seguintes histogramas, que são muito seme-

lhantes:

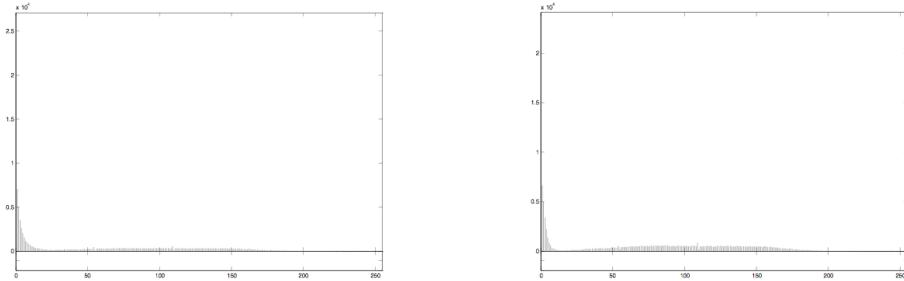


Figura 7.5: Histogramas.

A figura seguinte mostra a auto-correlação normalizada entre os pixels numa linha das respectivas imagens.

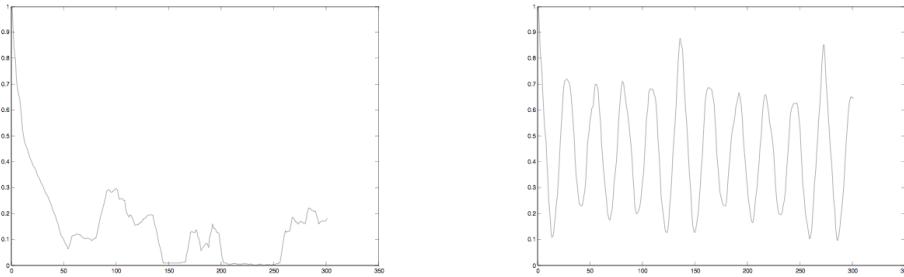


Figura 7.6: Auto-correlação normalizada entre os pixels.

A figura 7.6 direita mostra que os pixels vizinhos da imagem da direita da figura 7.4 exibem periodicamente uma auto-correlação muito elevada. Esta é facilmente explicada pela repetição periódica das barras verticais brancas na imagem da direita da figura 7.4.

Este exemplo será usado no decorrer desta apresentação para ilustrar as propriedades de de-correlação da codificação por transformadas.

### Codificação Diferencial ou preditiva

A diferença entre o valor actual e uma previsão desse valor é codificado. É bastante usado em casos em que amostras sucessivas do sinal não diferem muito, mas não são zero. Exemplo: vídeo, áudio.

Neste tipo de codificação apenas precisamos de codificar:

$$\Delta f(t_i) = f_{actual}(t_i) - f_{actual}(t_{i-1})$$

Se amostras sucessivas estão próximas umas de outras apenas precisamos de codificar a primeira amostra com um grande número de bits.

**Exemplo** Actual Data: 9 10 7 6

$\Delta f(t)$ : +9, +1, -3, -1.

Notemos que a transformação é sem perdas, visto que, a transformação inversa fornece uma reconstrução perfeita da imagem original.

### Transformadas usadas no processamento de imagem

Qualquer função (sinal) pode ser decomposto em componentes puramente sinusoidais (ondas seno de diferentes tamanhos/formas) as quais quando adicionadas formam o nosso sinal original. Na última década a Transformada Discreta de Cosenos (DCT) emergiu como a transformada na maioria dos sistemas de visualização. A DCT foi usada nos codificadores de imagem e vídeo standard, por exemplo, JPEG, MPEG, JVT, etc.

Algumas das transformadas usadas no processamento de imagens são por exemplo:

- KLT - Karhunen-Loève Transform;
- DFT - Discrete Fourier Transform;
- DCT - Discrete Cosine Transform;
- Wavelets.

**Nota** Relembremos: Propriedades matemáticas das transformadas.

- Transformação linear: definida por uma matriz real  $A_{n \times n} = (a_{ij})$

$$\begin{bmatrix} a_{0,0} & \dots & a_{0,N-1} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ a_{N-1,0} & \dots & a_{N-1,N-1} \end{bmatrix} \begin{bmatrix} x_0 \\ \vdots \\ \vdots \\ x_{N-1} \end{bmatrix} = \begin{bmatrix} c_0 \\ \vdots \\ \vdots \\ c_{N-1} \end{bmatrix}$$

- Ortonormalidade:  $A^{-1} = A^T$ .

A energia dos dados é igual à energia dos coeficientes:

$$\begin{aligned} \sum_{i=0}^{N-1} c_i^2 &= c^T c = (Ax)^T (Ax) = (x^T A^T)(AX) = x^T (A^T A)x = \\ &= x^T x = \sum_{i=0}^{N-1} x_i^2. \end{aligned}$$

Dizemos que o resultado de uma transformada são os coeficientes da transformada porque  $A^T c = x$ :

$$\begin{bmatrix} a_{0,0} & \dots & a_{N-1,0} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ a_{0,N-1} & \dots & a_{N-1,N-1} \end{bmatrix} \begin{bmatrix} c_0 \\ \vdots \\ \vdots \\ c_{N-1} \end{bmatrix} = \begin{bmatrix} x_0 \\ \vdots \\ \vdots \\ x_{N-1} \end{bmatrix}$$

$$\begin{bmatrix} a_{0,0} \\ \vdots \\ a_{0,N-1} \end{bmatrix} c_0 + \dots + \begin{bmatrix} a_{N-1,0} \\ \vdots \\ a_{N-1,N-1} \end{bmatrix} c_{N-1} = \begin{bmatrix} x_0 \\ \vdots \\ x_{N-1} \end{bmatrix}$$

*Vector Base 0*                    *Vector Base N - 1*

---

**Exemplo** A matriz A é ortonormal ( $A^2 = I \rightarrow A^{-1} = A; A^T = A = A^{-1}$ .), podemos ver a sua capacidade de compactação, quando a aplicamos a um sinal.

$$A = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix},$$

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} b \\ b \end{bmatrix} = \begin{bmatrix} \sqrt{2}b \\ 0 \end{bmatrix}.$$


---

### 7.2.3 Frequêcia espacial e a DCT

Como as restantes transformadas, a DCT tenta descorrelacionar os dados da imagem. Após a de-correlação cada coeficiente da transformada pode ser codificado independentemente sem perder a eficiência da compressão. Vamos agora descrever a DCT e as suas propriedades mais importantes.

A frequêcia espacial indica quantas vezes os valores dos pixels mudam ao longo de um bloco da imagem. A DCT formaliza esta noção com uma medida de quanto o conteúdo de uma imagem muda em correspondência com o número de ciclos de uma onda senoidal por bloco.

O papel da DCT é decompor o sinal original em componentes DC e AC. O coeficiente DC é, geralmente, o mais importante, enquanto os coeficientes AC, em geral, tornam-se menos importantes à medida que se distanciam do DC;

O papel da IDCT é reconstruir o sinal original.

### 7.2.4 Definição de DCT

Dada uma função de entrada  $f(i)$  a DCT transforma-a numa nova função  $F(u)$ , com os inteiros  $u$  com valores no mesmo intervalo que  $i$ . A definição da transformada 1D DCT é

$$F(u) = C(u) \sum_{i=0}^{M-1} f(i) \times \cos \frac{(2i+1)u\pi}{2M}. \quad (7.6)$$

onde  $i = u = 0, \dots, M - 1$ .

A definição da transformada inversa 1D IDCT:

$$f(i) = \sum_{u=0}^{M-1} C(u) \times F(u) \times \cos \frac{(2i+1)u\pi}{2M}. \quad (7.7)$$

onde  $i = u = 0, \dots, M - 1$ .

Com

$$C(u) = \begin{cases} \sqrt{\frac{1}{M}} & \text{se } u = 0, \\ \sqrt{\frac{2}{M}} & \text{noutro caso.} \end{cases} \quad (7.8)$$

É claro que para  $u = 0$   $F(u) = \sqrt{\frac{1}{M}} \sum_{i=0}^{M-1} f(x)$ . Consequentemente, o primeiro coeficiente da transformada corresponde ao valor médio da sequência tratada. Na literatura, este valor é referenciado como coeficiente DC. Todos os outros coeficientes são referenciados como coeficientes AC (os nomes surgem do uso histórico da DCT para análise de circuitos eléctricos com correntes directas e alternadas).

Para fixar ideias, ignoremos o  $f(i)$  e o  $C(u)$ . A representação do resultado para  $N = 8$  e variando o valor de  $u$  é mostrado na figura 7.7. De acordo com as observações anteriores, a primeira onda resulta num valor constante (DC), enquanto todas as outras fornecem ondas com frequências que crescem progressivamente.

Estas ondas são denominadas de funções cosseno de base. Note que estas funções de base são ortogonais, logo independentes, isto é, nenhuma das funções de base pode ser representada como combinação das outras funções de base.

Aqui, um ponto muito importante a notar é que em cada cálculo os valores dos pontos da função de base não mudarão. Somente os valores de  $f(i)$  mudarão em cada sub-sequência. Esta é uma propriedade muito importante, visto que mostra que as funções de base podem ser pre-cálculadas previamente e então ser multiplicadas com as sub-sequências. Isto reduz o número das operações matemáticas (isto é, multiplicações e adições) aumentando assim a eficiência da computação.

### 7.2.5 Definição de 2D DCT

Dada uma função de entrada  $f(i, j)$  sobre duas variáveis inteiras  $i$  e  $j$  (uma parte de uma imagem), a 2D DCT transforma-a numa nova função  $F(u, v)$ , com os inteiros  $u$  e  $v$  com valores no mesmo intervalo que  $i$  e  $j$ . A definição da transformada é:

$$F(u, v) = C(u)C(v) \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} f(i, j) \times \cos \frac{(2i+1)u\pi}{2M} \cos \frac{(2j+1)v\pi}{2N}. \quad (7.9)$$

onde  $i, u = 0, 1, \dots, M - 1; j = 1, 2, \dots, N - 1$ .

A transformada inversa é definida por:

$$f(i, j) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} C(u)C(v)F(u, v) \times \cos \frac{(2i+1)u\pi}{2M} \cos \frac{(2j+1)v\pi}{2N}. \quad (7.10)$$

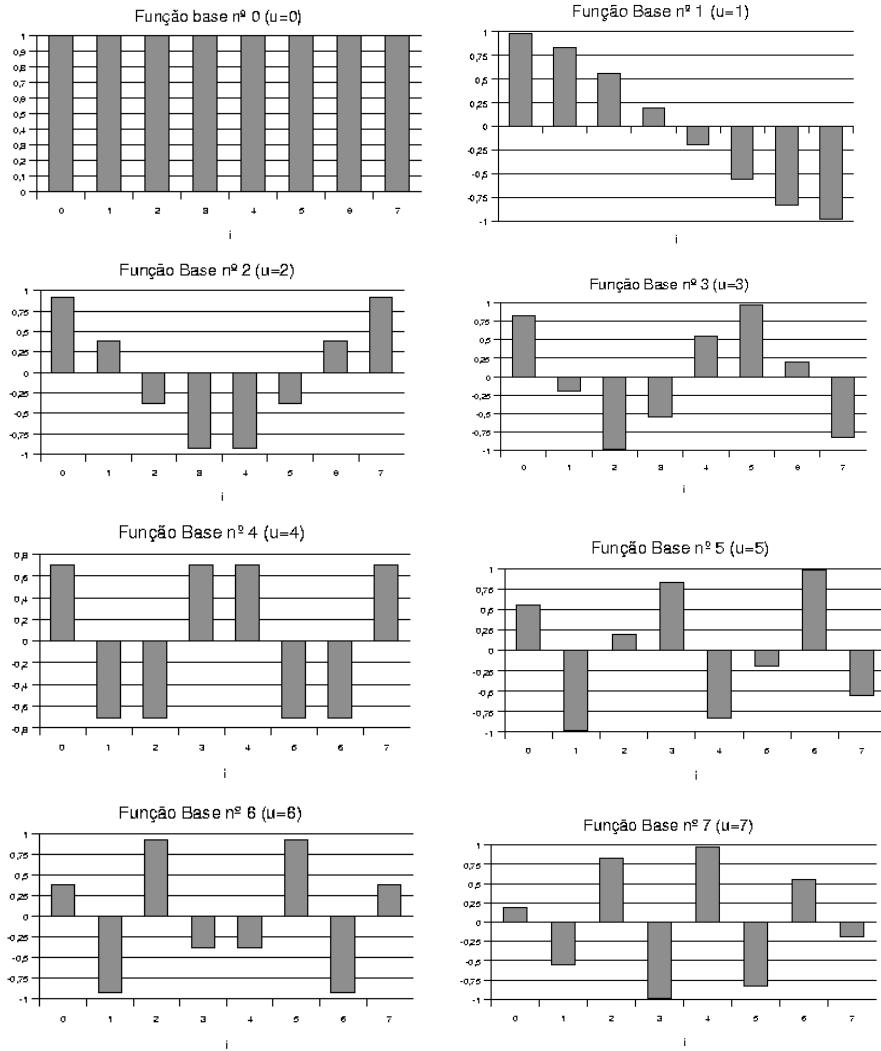


Figura 7.7: Funções de base da DCT.

A 2D DCT pode ser separada em duas 1D DCT:

$$G(i, v) = C(v) \sum_{j=0}^{N-1} f(i, j) \times \cos \frac{(2j+1)v\pi}{2N}. \quad (7.11)$$

$$F(u, v) = C(u) \sum_{i=0}^{M-1} G(i, v) \times \cos \frac{(2i+1)u\pi}{2M}. \quad (7.12)$$

onde  $i, u = 0, \dots, M-1; j = v = 0, \dots, N-1$ .

A 2D IDCT pode ser separada em duas 1D IDCT:

$$g(i, v) = \sum_{u=0}^{M-1} C(u) F(u, v) \times \cos \frac{(2i+1)u\pi}{2M}. \quad (7.13)$$

$$f(i, j) = \sum_{v=0}^{N-1} C(v) G(i, v) \times \cos \frac{(2j+1)v\pi}{2N}. \quad (7.14)$$

onde  $i, u = 0, \dots, M-1; j = v = 0, \dots, N-1$ .

Note na figura 7.8 que as funções de base exibem um crescimento progressivo da frequência em ambas as direções, horizontal e vertical.

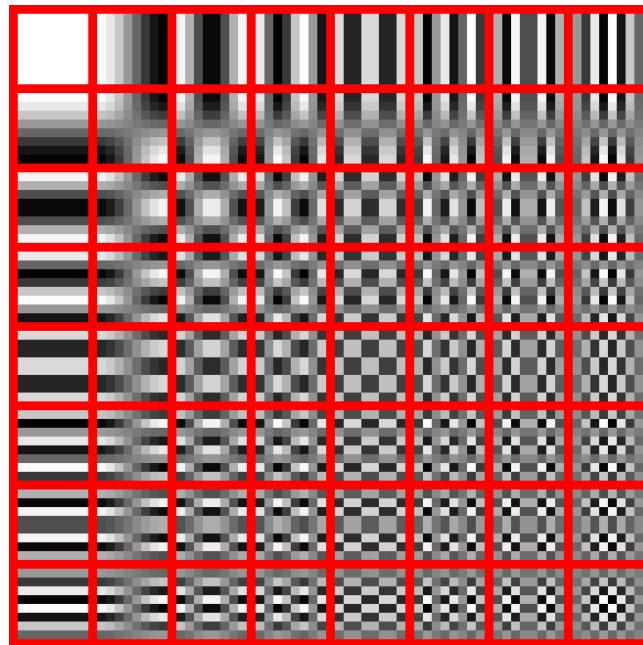


Figura 7.8: 2D DCT.

### 7.2.6 Propriedades da DCT

Vamos verificar algumas propriedades das DCT que têm um valor particular para as aplicações de processamento de imagem.

- Decorrelação
- Compactação de energia
- Separabilidade
- Simetria
- Ortogonalidade

#### Decorrelação

A principal vantagem da transformação de imagens é a remoção da redundância entre pixels vizinhos. Isto leva a coeficientes da transformada decorrelacionados os quais podem ser codificados independentemente.

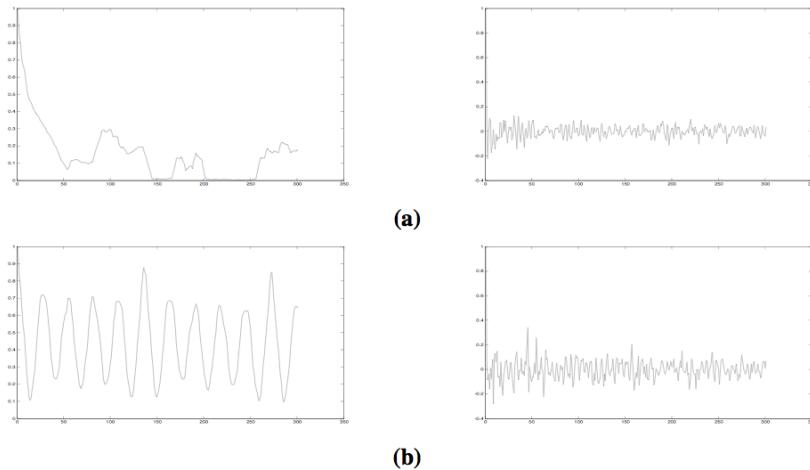


Figura 7.9: (a) Auto-correlação normalizada de uma imagem não correlacionada, antes e depois da DCT. (b) Auto-correlação normalizada de uma imagem correlacionada, antes e depois da DCT.

### Compactação de energia

A eficiência de um esquema de transformação pode ser regido directamente pela sua capacidade em compactar dados de entrada no menor número de coeficientes possível. Isto permite aos quantizadores eliminar coeficientes com amplitudes relativamente pequenas sem introduzir distorção visual na imagem reconstruída. A DCT exibe excelentes propriedades acumulação de energia para imagens altamente correlacionadas.

Vamos novamente considerar os dois exemplos das imagens mostradas inicialmente. Além das respectivas propriedades de correlação, a imagem não correlacionada possui mais variações bruscas de intensidades do que a imagem correlacionada. Assim, a primeira tem mais altas frequências que a outra. Claramente, a imagem não correlacionada tem a sua energia mais espalhada do que a correlacionada cuja energia está compactada na região das baixas frequências (i.e., região superior esquerda).

As figuras 7.11 (a) e (b) contêm uma grande área de variações suaves de intensidades. Estas imagens podem ser classificadas como imagens de baixas frequências com poucos detalhes espaciais. Uma operação DCT nestas imagens fornece uma muito boa acumulação de energia na região de baixa frequências da imagem transformada.

A figura 7.11 (c) contém um número de limites (variações bruscas de intensidades) e consequentemente pode ser classificada como uma imagem de altas frequências com baixo conteúdo espacial. No entanto a imagem possui alta correlação, a qual é explorada pelo algoritmos DCT para fornecer boa acumulação de energia.

A figura 7.12 (d) e (e) são imagens com progressivamente alta frequência e conteúdo espacial. Consequentemente, os coeficientes da transformada estão espalhados por todas as baixas e altas frequências.

A figura 7.11 (e) mostra periodicidade, logo a DCT contém impulsos com

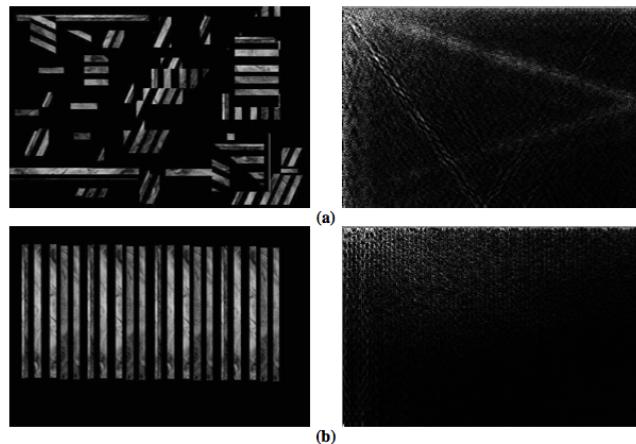


Figura 7.10: (a) Imagem não correlacionada e a sua DCT. (b) Imagem correlacionada e a sua DCT.

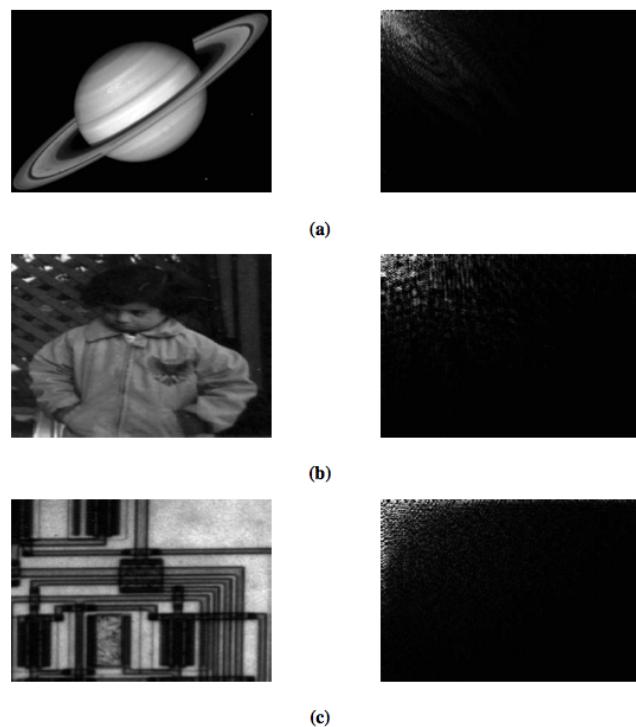


Figura 7.11: (a) Imagem "Saturn"e a sua DCT; (b) Imagem "Child"e a sua DCT; (c) Imagem "Circuit"e a sua DCT.

amplitudes proporcionais ao peso de uma frequência em particular na "waveform"original.

Da discussão anterior pode ser inferido que a DCT consegue excelente acumulação de energias para imagens correlacionadas.

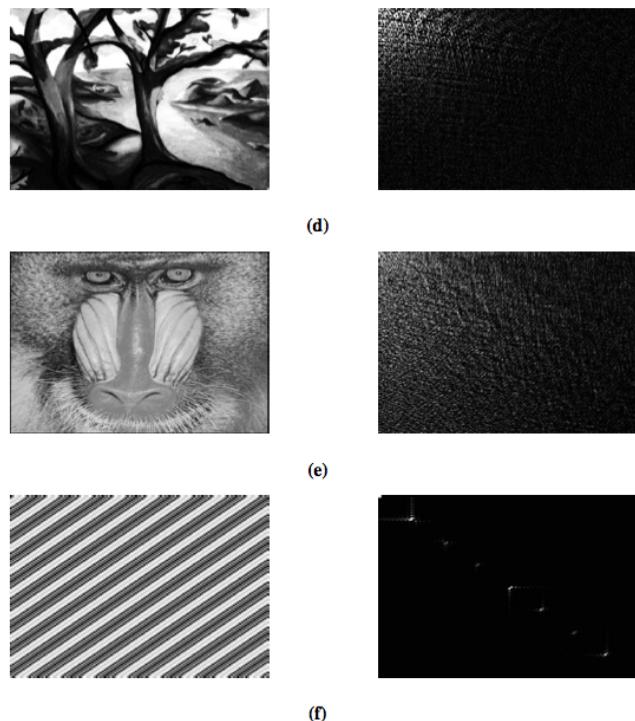


Figura 7.12: (d) Imagem "Trees" e a sua DCT; (e) Imagem "Baboon" e a sua DCT; (f) Imagem da onda seno e a sua DCT.

### Separabilidade

Esta propriedade, conhecida como separabilidade, tem a principal vantagem de possibilitar o cálculo de  $F(u, v)$  em dois passos usando sucessivamente operações 1D sobre as linhas e as colunas de uma imagem. Os argumentos aplicados podem ser igualmente aplicados para o cálculo da DCT inversa.

### Simetria

Se olhar bem para a equação da transformada DCT sobre as linhas e sobre as colunas, repare que as operações são funcionalmente idênticas. Este tipo de transformada é denominada transformada simétrica.

Esta propriedade é extremamente útil visto que esta implica que a matriz transformação pode ser calculada previamente e depois aplicada à imagem aumentando assim a eficiência da computação.

### Ortogonalidade

Como vimos anteriormente, as funções de base da DCT são ortogonais. Consequentemente, a matriz da transformada inversa  $A$  é igual à sua transposta i.e.  $A^{-1} = A^T$ . Assim, e em adição às suas características de decorrelação, esta propriedade permite redução na complexidade da pré-computação.

### 7.2.7 DCT versus DFT/KLT

A Transformada KLT ( Karhunen-Loeve Transform) é uma transformada linear onde as funções base são tomadas das propriedades estatísticas dos dados da imagem, e pode assim ser adaptativa. É óptima no sentido de compactação de energia. No entanto a transformada KLT é geralmente não separável, e por isso a multiplicação total da matriz tem de ser efectuada. A derivação das bases respectivas para cada sub-bloco da imagem requer uma quantidade não razoável de recursos computacionais. Apesar de terem sido já propostos alguns algoritmos rápidos da KLT a complexidade global da KLT é significativamente superior do que os respectivos algoritmos da DCT ou DFT.

A transformada DFT (Discrete Fourier Transform) é linear, separável e simétrica. Assim, tal como a DCT, tem funções de base fixas e implementações rápidas são possíveis. Também possui boas características de decorrelação e de compactação de energia. No entanto, a DFT é uma transformada complexa e por isso estipula que a magnitude e a fase da informação terão de ser codificadas. Em adição existem estudos que mostraram que a DCT fornece melhor compactação de energia do que a DFT para a maior parte das imagens naturais. Além disso, a periodicidade implícita da DFT tem como consequência descontinuidades nas fronteiras que resultam num conteúdo significativo de altas frequências. Depois da quantização, o fenómeno de Gibbs provoca que nos pontos limites podem ser tomados valores errados. Pode encontrar mais informação sobre o fenómeno de Gibbs em (<http://euler.us.es/plopez/fenomeno-de-Gibbs.htm>).

Comparando a DCT com a DFT, a DCT é uma transformada que apenas envolve a parte real da DFT.

### 7.2.8 DCT e quantização

As figuras 7.13 a 7.18 mostram as imagens reconstruídas efectuando a DCT inversa dos coeficientes quantizados. Nitidamente a DCT(25%) introduz efeito de "blurring" em todas as imagens visto que apenas um quarto do número total de coeficientes é usado para a reconstrução. O resultado nas imagens Trees pode ser explicado pelo facto de a imagem possuir muitos detalhes de alta frequência não descorrelacionados. Consequentemente, eliminando os coeficientes da DCT de alta frequência resulta numa perda de qualidade. A figura senos é facilmente explicada se se examinar a sua DCT. A remoção dos coeficientes de alta frequência resulta na remoção de certas frequências que estão originalmente presentes na onda do seno. Depois de se perder certas frequências não é possível a reconstrução perfeita. No entanto a DCT(75%) fornece uma reconstrução excelente para todas as imagens excepto para a onda seno.

### 7.2.9 Wavelets

#### Transformada discreta de wavelet (DWT)

Uma forma de separar as variações suaves dos detalhes é decompondo a imagem usando a "Discrete Wavelet Transform"(DWT).

O procedimento é o seguinte. Um filtro "low-pass" e um "filtro high-pass" são escolhidos, de tal forma que eles dividem exactamente a escala da frequência entre eles. Este par de filtros são denominados filtros de análise. Primeiro o filtro "low-pass" é aplicado para cada linha dos dados, obtendo assim os componentes

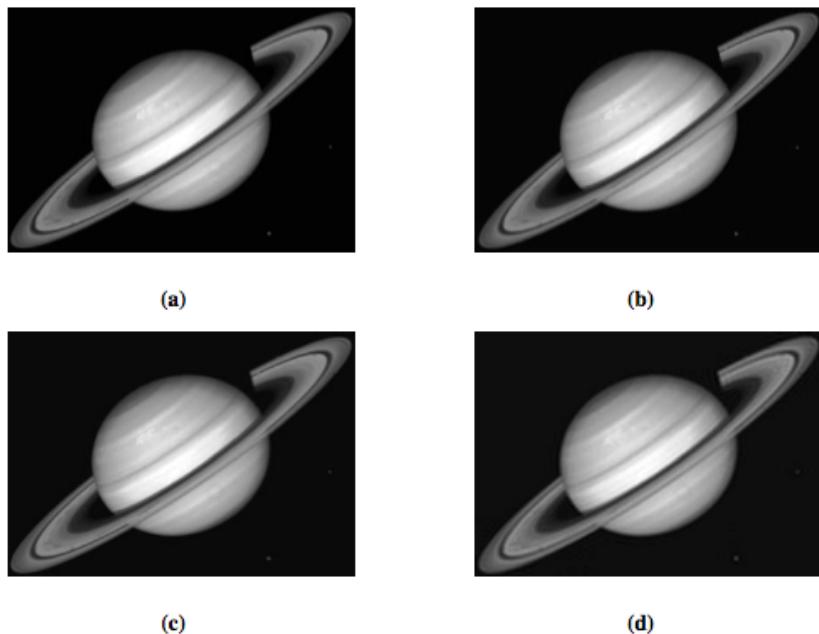


Figura 7.13: DCT inversa da imagem "Saturn"; (a) 100%; (b) 75%; (c) 50%; (d) 25%.

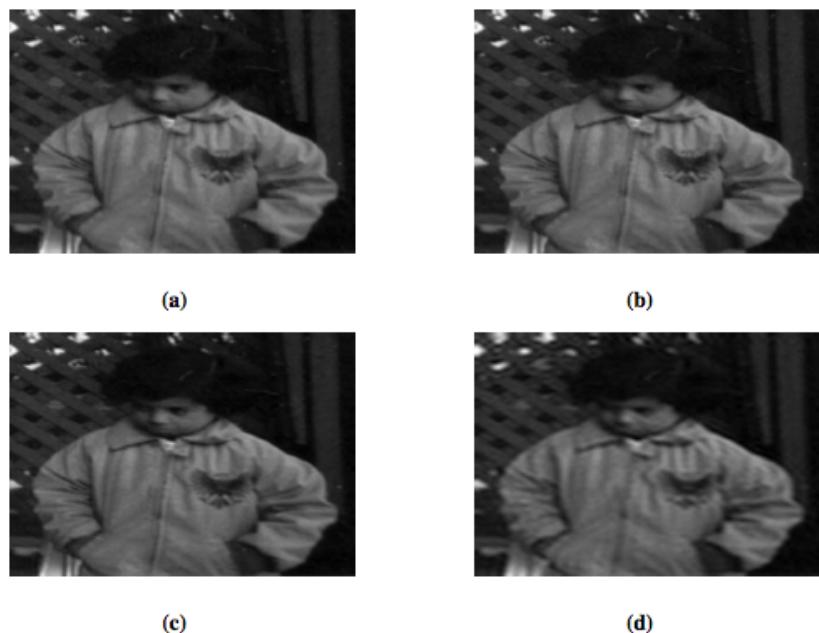


Figura 7.14: DCT inversa da imagem "Child"; (a) 100%; (b) 75%; (c) 50%; (d) 25%.

de baixa frequência da linha. Mas, visto que o filtro "low-pass" é um filtro "half

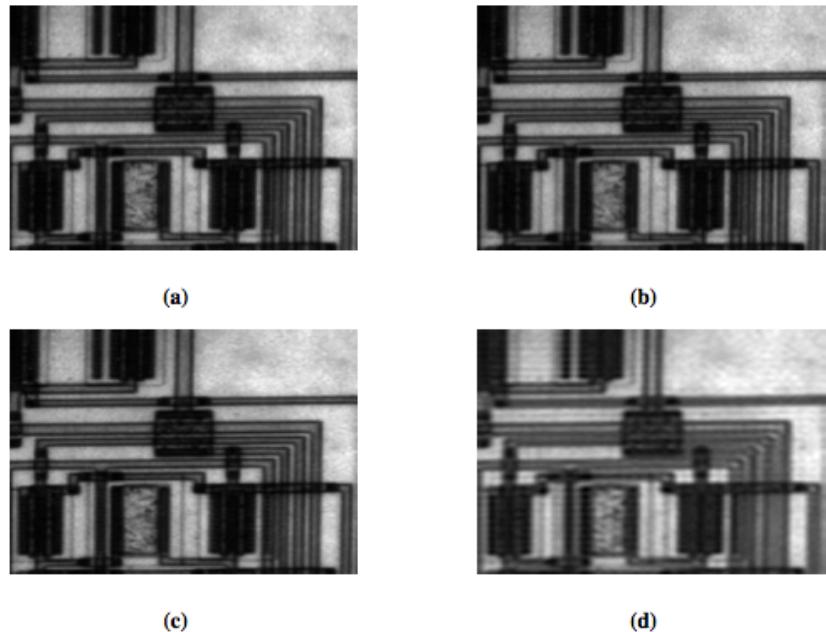


Figura 7.15: DCT inversa da imagem "Circuit"; (a) 100%; (b) 75%; (c) 50%; (d) 25%.

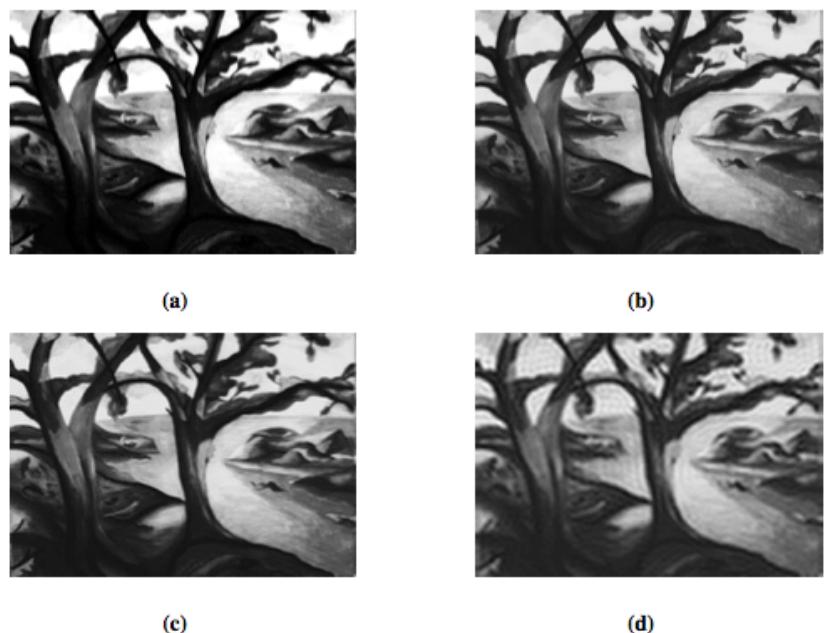


Figura 7.16: DCT inversa da imagem "Trees"; (a) 100%; (b) 75%; (c) 50%; (d) 25%.

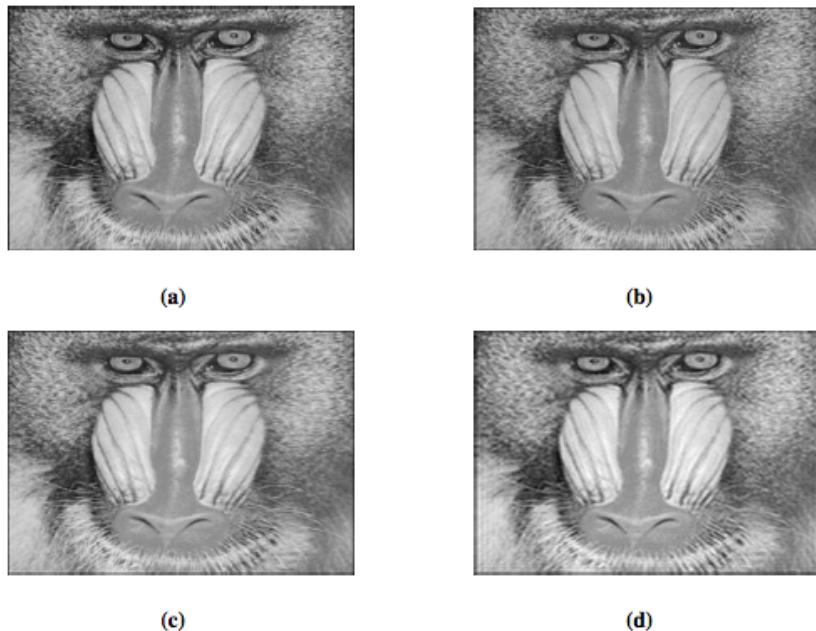


Figura 7.17: DCT inversa da imagem "Baboon"; (a) 100%; (b) 75%; (c) 50%; (d) 25%.

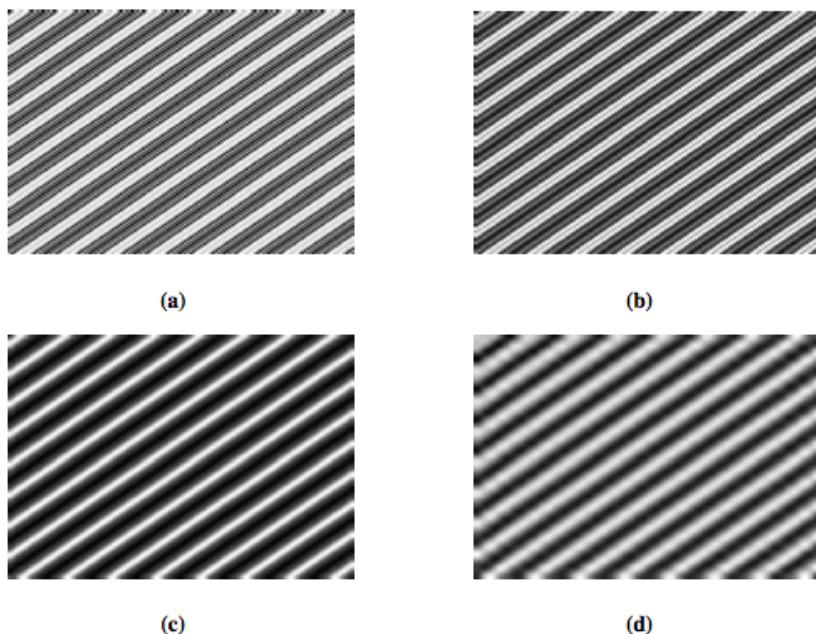


Figura 7.18: DCT inversa da onda seno; (a) DCT(100%); (b) DCT(75%); (c) DCT(50%); (d) DCT(25%).



Figura 7.19: Flor Original e com 16, 8 e 4 coeficientes de DCT.

band”, os dados de saída contêm frequências apenas na primeira metade do domínio de frequências original. Assim, pelo Teorema da amostragem de Shannon, eles podem ser sub-amostrados para metade, de tal forma que os dados de saída contêm apenas metade do número de amostras originais.

De seguida o filtro “high-pass” é aplicado aos mesmos dados da linha e de forma similar os componentes “high pass” são separados e colocados ao lado dos componentes “low-pass”. Este procedimento é efectuado em todas as linhas.

Finalmente, a filtragem é efectuada para cada coluna dos dados intermédios. O array bi-dimensional de coeficientes resultante contém quatro blocos de dados, cada um denominado LL (low-low), HL (high-low), LH (low-high) and HH (high-high). A sub-banda LL pode ser decomposta de novo, da mesma forma produzindo ainda mais sub-bandas.

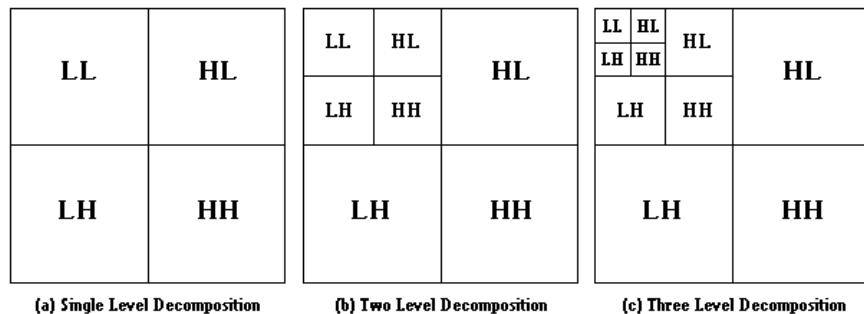


Figura 7.20: Decomposição piramidal de uma imagem.

Como visto anteriormente, a sub-banda LL do nível mais elevado pode ser considerada a mais importante, e as outras sub-bandas de detalhes podem ser classificadas de menor importância, com o nível de importância a decrescer do topo da pirâmide até às sub-bandas de baixo.

#### **Transformada discreta de wavelet inversa (IDWT)**

A transformada inversa é usada para reconstruir a imagem. Um par de filtros low-pass e high-pass são também usados aqui. Estes filtros são denominados filtros de síntese. O procedimento de filtragem é exactamente o inverso - começamos pelo nível mais elevado e aplicamos os filtros primeiro às colunas e depois às linhas, e pressegue-se no nível seguinte até chegar ao nível 1.

Alguns links interessantes sobre DWT:

[www.wavelet.org](http://www.wavelet.org)  
[www-stat.stanford.edu/~wavelab/](http://www-stat.stanford.edu/~wavelab/)  
[www.mathsoft.com/wavelets.html](http://www.mathsoft.com/wavelets.html)

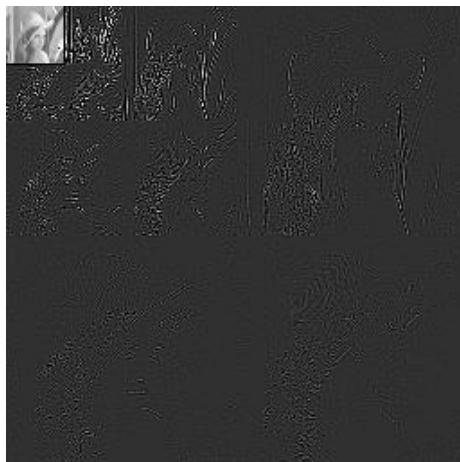


Figura 7.21: Decomposição da imagem "Lena" em três níveis.

#### **7.2.10 Quantização**

A quantização refere-se ao processo de aproximar o conjunto contínuo de valores nos dados da imagem com um conjunto de valores finito (de preferência pequeno). O input do quantizador são os dados originais, e o output é sempre um entre um número finito de níveis. O quantizador é uma função cujo conjunto de valores de output é discreto e usualmente finito. Obviamente, este é um processo de aproximação, e um bom quantizador é um que representa o sinal original com um mínimo de perdas ou distorção.

Existem dois tipos de quantização - quantização escalar e quantização vectorial. Na quantização escalar, cada símbolo de input é tratado separadamente na produção do output, enquanto que na quantização vectorial os símbolos de entrada são processados em grupos chamados vectores para gerar o output. A

quantização vectorial optimiza o processo de quantização mas aumenta a complexidade computacional.

Vamos ver um pouco como funciona a quantização escalar.

Um quantizador pode ser especificado pelas suas partições do input e os níveis do output. Se o input é dividido em níveis igualmente espaçados, então o quantizador é denominado Quantizador Uniforme, senão é denominado Quantizador Não Uniforme. Um quantizador uniforme pode ser especificado pelo seu limite inferior e tamanho do passo. A implementação de um quantizador uniforme é mais simples do que a de um quantizador não uniforme.

### 7.2.11 Quantização escalar vs. vectorial

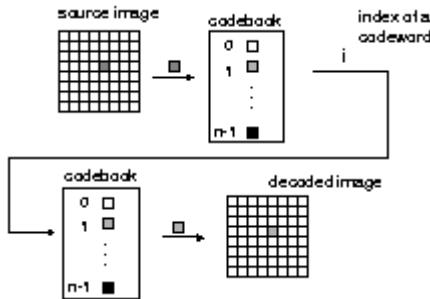


Figura 7.22: Quantização escalar.

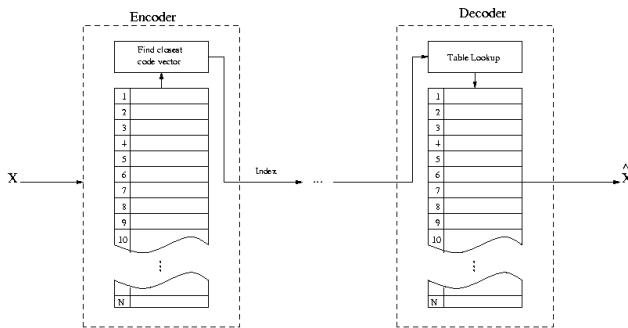


Figura 7.23: Quantização vectorial.

### 7.2.12 Quantização escalar

A quantização escalar (SQ) é o mais simples de todos os esquemas de compressão com perdas. Pode ser descrito como uma função que mapeia cada elemento de um subconjunto de  $\mathbb{R}$  a um valor particular dentro desse subconjunto. Considere particionar  $\mathbb{R}$  em  $M$  intervalos disjuntos

$$I_q = [t_q, t_{q+1}), q = 0, 1, \dots, M - 1$$

com

$$-\infty = t_0 < t_1 < \dots < t_M = +\infty.$$

Dentro de cada intervalo, um ponto  $\hat{x}$  é selecionado como o valor de saída (ou palavra código) de  $I_q$ . Um quantizador escalar é assim um mapeamento de  $\mathbb{R}$  a  $0, 1, \dots, M - 1$ . Especialmente, para um dado  $x$ ,  $Q(x)$  é o índice  $q$  do intervalo  $I_q$  que contém  $x$ . O dequantizador é dado por

$$\overline{Q^{-1}}(q) = \hat{x}_q.$$

**Exemplo** Seja  $M = 4, t_1 = -1, t_2 = 0, t_3 = 1, \hat{x}_0 = -1.5, \hat{x}_1 = -0.5, \hat{x}_2 = 0.5, \hat{x}_3 = 1.5$ . Então, se  $x < -1$ , a versão quantizada de  $x$  é  $-1.5$  (índice = 0). Especificamente,  $Q(x) = 0$  e  $\overline{Q^{-1}}(Q(x)) = Q^{-1}(0) = -1.5$ . Da mesma forma, se  $0 \leq x < 1$ , a versão quantizada de  $x$  é  $0.5$  (índice = 2). Esta situação é ilustrada na figura 7.24

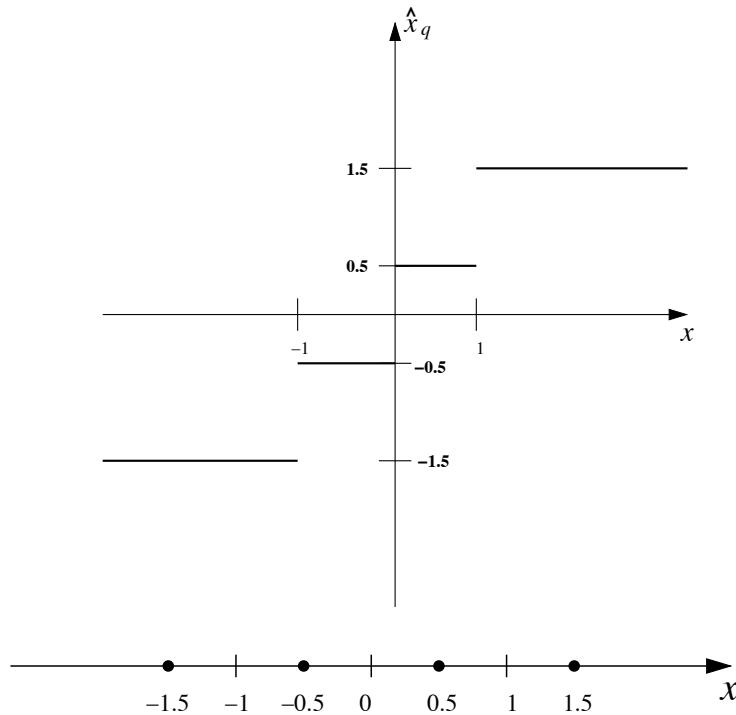


Figura 7.24: Duas representações diferentes do mesmo quantizador escalar.



Figura 7.25: O caso mais geral de quantização escalar.

A figura 7.25 representa o caso geral da quantização escalar. Esta figura mostra que quando  $x \in I_q = [t_q, t_{q+1})$ , então  $\overline{Q^{-1}}(Q(x)) = \overline{Q^{-1}}(q) = \hat{x}$ . Claramente, o  $t_q$  pode ser visto como limites, ou limites de decisão, para o  $\hat{x}_q$ .

### Quantização escalar uniforme

Um quantizador escalar uniforme partitiona o domínio dos valores de entrada em intervalos igualmente espaçados, excepto possivelmente nos dois intervalos exteriores.

O valor de saída correspondente a cada intervalo é o ponto médio do intervalo ou então o centróide do intervalo ( $\text{round}(\sum xp_x)$ ).

O tamanho de cada intervalo é chamado tamanho do passo e denotado por  $\Delta$ .

Existem dois tipos de quantizadores escalares uniformes:

- Quantizadores midrise possuem número par de níveis de saída;
- Quantizadores midtread possuem número ímpar de níveis de saída, incluindo o zero como um deles.

Para o caso especial em que  $\Delta = 1$ , podemos simplesmente calcular os valores de saída do quantizador como:

$$Q_{\text{midrise}}(x) = \lceil x \rceil - 0.5$$

$$Q_{\text{midtread}}(x) = \lfloor x + 0.5 \rfloor$$

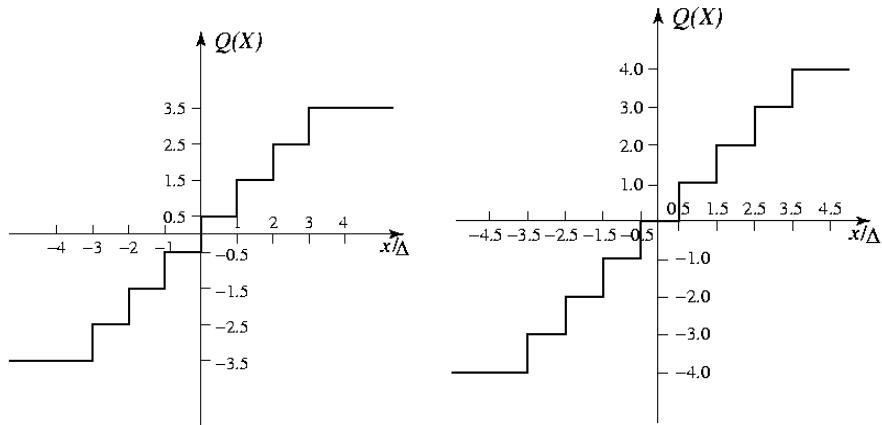


Figura 7.26: Quantização escalar "Midrise" vs "Midtread".

### 7.2.13 Quantização não uniforme

Na quantização escalar não uniforme, para zonas com maior quantidade de valores são definidos passos de quantização menores, enquanto que, para zonas com menor quantidade de valores são definidos passos de quantização maiores.

O Algoritmo de Loyd (1957) tem por objectivo encontrar os intervalos que implicam o menor erro de quantização.

Prova-se que os intervalos e níveis de quantificação são dados de forma óptima através das condições de Lloyd-Max que se escrevem

$$t_q = \frac{\hat{x}_{q-1} + \hat{x}_q}{2} \quad q = 1, 2, \dots, M - 1. \quad (7.15)$$

$$\hat{x}_q = \frac{\int_{t_q}^{t_{q+1}} x f_X(x) dx}{\int_{t_q}^{t_{q+1}} f_X(x) dx} \quad q = 0, 1, \dots, M - 1. \quad (7.16)$$

ou

$$\hat{x}_q = \text{round} \left( \frac{\sum_{x=t_q}^{t_{q+1}} x p_X(x)}{\sum_{x=t_q}^{t_{q+1}} p_X(x)} \right) \quad q = 0, 1, \dots, M - 1. \quad (7.17)$$

de onde podemos concluir que os níveis de quantificação óptimos são dados pelos centroides e os limites de quantificação de cada região são dados pelos pontos médios entre cada par de centroides. A resolução dos sistema de equações (7.15) e (7.16) faz-se de modo iterativo partindo de um conjunto inicial de centroides e calculando iterativamente a distorção, até que não haja uma diminuição significativa dessa distorção entre um ponto e o seguinte, por exemplo, até que:

$$\frac{1}{M} \sum_{i=1}^M (\hat{x}_i^n - \hat{x}_i^{n-1})^2 < \epsilon \quad (7.18)$$

sendo o  $n$  o número da iteração actual e  $n - 1$  o número da iteração anterior.

Em resumo. Uma estratégia para encontrar os intervalos de quantização e os níveis de reconstrução consiste em minimizar a variância do erro de quantização (Erro quadrático médio). Esta é a estratégia do Quantizador de Lloyd-Max.

## 7.3 A norma JPEG

JPEG é um standard de compressão desenvolvido pela “Joint Photographic Experts Group” que foi formalmente aceite como standard internacional em 1992. Este é um método de compressão com perdas, de imagens que usa codificação por transformada - DCT.

### 7.3.1 Modos de operação

- Codificação sequencial: cada componente da imagem é codificado numa única passagem da esquerda para a direita e de cima para baixo;
- Codificação progressiva: A imagem é codificada em várias passagens para aplicações em que o tempo de transmissão é longo e o utilizador prefere ver construir a imagem de mais grosseira para mais clara;
- Codificação sem perdas;

- Codificação hierárquica: a imagem é codificada com múltiplas resoluções, de forma a que uma versão com menor resolução possa ser acedida antes de descomprimir a imagem na sua resolução final.

Cada um destes modos possui um ou mais codecs. Os diferentes codecs diferem de acordo com a precisão de imagem que eles suportam ou com o método de codificação entrópica que eles usam.

Muitas implementações possuem apenas alguns destes modos ou mesmo só parte do codec (codificador ou descodificador). A maior parte das implementações do mercado possui apenas o codec sequencial de base ("baseline"). É fundamentalmente deste que vamos falar agora.

### 7.3.2 Principais passos da compressão JPEG

Na figura 7.27 podemos ver os passos principais da compressão JPEG.

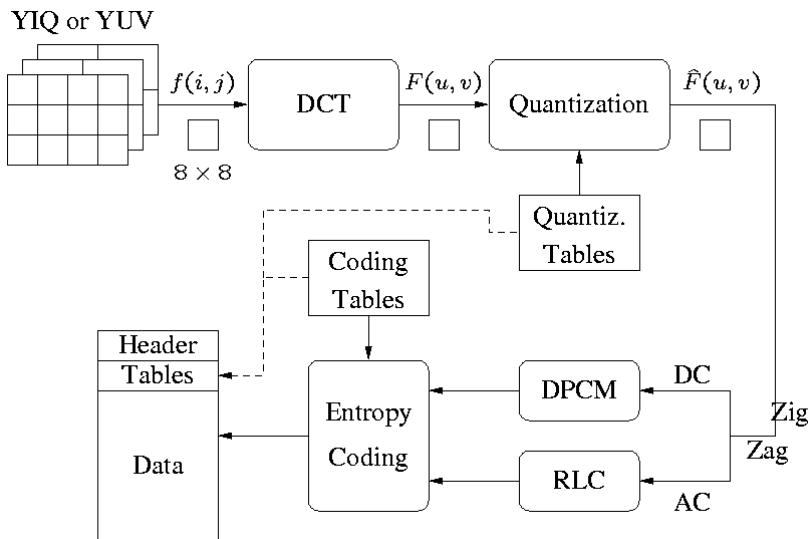


Figura 7.27: Esquema da codificação JPEG.

Os principais passos da compressão JPEG são:

- Transformação de cor: o objectivo deste passo é o de transformar as imagens representadas em RGB em YIQ ou YUV. De seguida pode-se fazer uma sub - amostragem dos planos de cor;
- Transformação espacial: neste passo é xecutada a DCT em blocos da imagem (para imagens em 8 ou 12 bits - o modo de base apenas executa a DCT para 8);
- Quantização;
- Ordenação em zig-zag e codificação run-length;
- Codificação entrópica (ou estatística).

De seguida vamos especificar cada um destes passos.

### 7.3.3 DCT em blocos

Cada imagem é dividida em blocos de  $8 \times 8$ . A 2D DCT é aplicada a cada um desses blocos. O facto de usar blocos neste passo cria o chamado efeito de bloco quando a taxa de compressão especificada pelo utilizador é elevada.

No caso JPEG a DCT é aplicada a blocos de 8. Usando a 2D DCT separável teremos:

$$G(i, v) = C(v) \sum_{j=0}^7 f(i, j) \cos \frac{(2j+1)v\pi}{16}, \quad (7.19)$$

$$F(u, v) = C(u) \sum_{i=0}^7 G(i, v) \cos \frac{(2i+1)u\pi}{16}. \quad (7.20)$$

onde  $i, u = 0, \dots, 7; j, v = 0, \dots, 7$  e com

$$C(u) = \begin{cases} \sqrt{\frac{1}{8}} & \text{se } u = 0, \\ \sqrt{\frac{2}{8}} & \text{noutro caso.} \end{cases} \quad (7.21)$$

As inversas serão:

$$G(i, v) = \sum_{u=0}^7 C(u) F(u, v) \cos \frac{(2i+1)u\pi}{16}, \quad (7.22)$$

$$f(i, j) = \sum_{v=0}^7 C(v) G(i, v) \cos \frac{(2j+1)v\pi}{16}. \quad (7.23)$$

### 7.3.4 Quantização

Depois de efectuada a transformada cada conjunto de 64 coeficientes é quantizador utilizando tabelas de quantização. Os valores destas tabelas são valores inteiros que definem o passo de quantização.

A quantização corresponde à divisão inteira de cada coeficiente pelo correspondente na tabela de quantização.

$$F^Q(u, v) = \text{Integer Round} \frac{F(u, v)}{Q(u, v)}$$

As tabelas usadas podem ser as dos standard ou indicadas pelo utilizador; As mesmas tabelas de quantização são usadas para codificar todas as amostras de um componente.

A ordem de crescimento dos valores da tabela é inversa à ordem de importância dos coeficientes DCT. Isto significa que vamos dividir pelos maiores valores os coeficientes julgados menos importantes.

Os valores que constituem a tabela de quantização determinam a qualidade final da compressão. Com valores pequenos vamos obter uma taxa de compressão pequena, com grandes valores a taxa de compressão será grande, mas a qualidade vai diminuir.

Após quantização alguns valores (especialmente o zero) têm frequências de ocorrência elevada. Para aumentar mais a taxa de compressão vamos explorar esta redundância estatística usando algoritmos de codificação entrópica.

Com já mencionamos anteriormente, a DCT e a quantização têm papéis muito importantes na qualidade final de uma imagem comprimida segundo o algoritmo JPEG. Vamos ver nas imagens seguintes o que acontece quando efectuamos a DCT e a quantização a um bloco de uma imagem. Repare como o comportamento da DCT difere quando trata blocos com pequenas variações e blocos com grandes variações. Esta diferença reflecte-se no erro do bloco reconstruído em relação ao bloco original, e claro na qualidade do resultado da reconstituição da imagem comprimida.

### 7.3.5 Funcionamento da DCT e da Quantização

De seguida vamos apresentar o funcionamento do codec JPEG em dois blocos distintos retirados da imagem Lena. Pode ver na figura 7.28 que o primeiro bloco é um bloco mais suave enquanto que no segundo bloco já existem variações um pouco mais bruscas .



Figura 7.28: Imagem Lena com a representação de dois blocos de  $8 \times 8$ .

**Exemplo** Neste primeiro exemplo partimos do bloco mais suave que está representado na tabela 7.1. Este é um bloco muito suave, onde não existem variações bruscas dos coeficientes.

Antes de efectuar a DCT do bloco os valores da imagem são deslocados de forma a ficarem centrados em zero. Como neste caso o intervalo inicial é  $[0, 255]$  para centrar em zero teremos de subtrair a todos os elementos o valor 128. Obtemos assim o bloco 7.2.

200	202	189	188	189	175	175	175
200	203	198	188	189	182	178	175
203	200	200	195	200	187	185	175
200	200	200	200	197	187	187	187
200	205	200	200	195	188	187	175
200	200	200	200	200	190	187	175
205	200	199	200	191	187	187	175
210	200	200	200	188	185	187	186

Tabela 7.1: Bloco original da imagem Lena.pgm.

72	74	61	60	61	47	47	47
72	75	70	60	61	54	50	47
75	72	72	67	72	59	57	47
72	72	72	72	69	59	59	59
72	77	72	72	67	60	59	47
72	72	72	72	72	62	59	47
77	72	71	72	63	59	59	47
82	72	72	72	60	57	59	58

Tabela 7.2: Bloco original da imagem Lena.pgm representados no intervalo [-128,127] -  $f(i,j)$ .

Efectuando a DCT separável cujas formulas foram apresentadas em 7.20 obtemos o bloco de coeficientes DCT apresentado em 7.3. Note que os valores de maior energia encontram-se no canto superior esquerdo.

515	65	-12	4	1	2	-8	5
-16	3	2	0	0	-11	-2	3
-12	6	11	-1	3	0	1	-2
-8	3	-4	2	-2	-3	-5	-2
0	-2	7	-5	4	0	-1	-4
0	-3	-1	0	4	1	-1	0
3	-2	-3	3	3	-1	-1	3
-2	5	-2	4	-2	2	-3	0

Tabela 7.3: DCT do bloco da tabela 7.2 -  $F(u,v)$ 

Usando a tabela de quantização típica do standard JPEG, apresentada na tabela 7.4, nos coeficientes DCT do bloco apresentado na tabela 7.3 obtemos o bloco apresentado na tabela 7.5

Procedendo agora à quantização inversa e à DCT inversa, obtemos respectivamente as tabelas 7.6 e 7.7.

Finalmente colocando novamente os valores no intervalo [0,255] obtemos a tabela 7.8. O erro de final é apresentado na tabela 7.9.

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	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Tabela 7.4: Tabela de quantização típica do standard JPEG - Q(i,j)

32	6	-1	0	0	0	0	0
-1	0	0	0	0	0	0	0
-1	0	1	0	0	0	0	0
-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	0	0	0

Tabela 7.5: Quantização dos coeficientes DCT 7.3 - F(u,v)/Q(u,v).

512	66	-10	0	0	0	0	0
-12	0	0	0	0	0	0	0
-14	0	16	0	0	0	0	0
-14	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

Tabela 7.6: Quantização inversa dos coeficientes DCT quantizados 7.5 - Q(i,j) \* (F(u,v)/Q(i,j)).

71	68	63	58	54	50	49	48
73	71	68	64	60	55	52	50
75	75	74	72	67	61	55	52
74	75	76	75	70	63	55	51
72	73	74	73	68	61	54	49
72	72	71	69	64	58	53	49
76	74	71	67	62	58	55	53
79	76	72	66	62	59	57	56

Tabela 7.7: DCT inversa dos coeficientes 7.6 -  $\hat{f}(i, j)$ .

199	196	191	186	182	178	177	176
201	199	196	192	188	183	180	178
203	203	202	200	195	189	183	180
202	203	204	203	198	191	183	179
200	201	202	201	196	189	182	177
200	200	199	197	192	186	181	177
204	202	199	195	190	186	183	181
207	204	200	194	190	187	185	184

Tabela 7.8: Valores obtidos depois do processo de compressão/descompressão do bloco 7.1 -  $\hat{f}(i, j) + 128$ .

1	6	-2	2	7	-3	-2	-1
-1	4	2	-4	1	-1	-2	-3
0	-3	-2	-5	5	-2	2	-5
-2	-3	-4	-3	-1	-4	4	8
0	4	-2	-1	-1	-1	5	-2
0	0	1	3	8	4	6	-2
1	-2	0	5	1	1	4	-6
3	-4	0	6	-2	-2	2	2

Tabela 7.9: Erro entre os valores originais 7.1 e os finais 7.8 -  $f(i, j) - \hat{f}(i, j)$ .

**Exemplo** O bloco, cujas transformações estão representadas nas imagens seguintes é um bloco onde já existem variações bruscas dos coeficientes. Repare no resultado da DCT e como este se reflecte depois no erro final.

70	70	100	70	87	87	150	187
85	100	96	79	87	154	87	113
100	85	116	79	70	87	86	196
136	69	87	200	79	71	117	96
161	70	87	200	103	71	96	113
161	123	147	133	113	113	85	161
146	147	175	100	103	103	163	187
156	146	189	70	113	161	163	197

Tabela 7.10: Bloco original da imagem Lena.pgm.

Antes de efectuar a DCT do bloco os valores da imagem são deslocados de forma a ficarem centrados em zero. Como neste caso o intervalo inicial é  $[0, 255]$  para centrar em zero teremos de subtrair a todos os elementos o valor 128. Obtemos assim o bloco 7.2.

Efectuando a DCT separável cujas fórmulas foram apresentadas em 7.20 obtemos o bloco de coeficientes DCT apresentado em 7.12. Note que os valores de maior energia encontram-se no canto superior esquerdo.

-58	-58	-28	-58	-41	-41	22	59
-43	-28	-32	-49	-41	26	-41	-15
-28	-43	-12	-49	-58	-41	-42	68
8	-59	-41	72	-49	-57	-11	-32
33	-58	-41	72	-25	-57	-32	-15
33	-5	19	5	-15	-15	-43	33
18	19	47	-28	-25	-25	35	59
28	18	61	-58	-15	33	35	69

Tabela 7.11: Bloco original da imagem Lena.pgm representados no intervalo [-128,127] -  $f(i,j)$ .

-80	-40	89	-73	44	32	53	-3
-135	-59	-26	6	14	-3	-13	-28
47	-76	66	-3	-108	-78	33	59
-2	10	-18	0	33	11	-21	1
-1	-9	-22	8	32	65	-36	-1
5	-20	28	-46	3	24	-30	24
6	-20	37	-28	12	-35	33	17
-5	-23	33	-30	17	-5	-4	20

Tabela 7.12: DCT do bloco da tabela 7.2 -  $F(u,v)$

Usando a tabela de quantização típica do standard JPEG apresentada na tabela 7.4, nos coeficientes DCT do bloco apresentado na tabela 7.12 obtemos o bloco apresentado na tabela tab:JPEG-DCT13

-5	-4	9	-5	2	1	1	0
-11	-5	-2	0	1	0	0	-1
3	-6	4	0	-3	-1	0	1
0	1	-1	0	1	0	0	0
0	0	-1	0	0	1	0	0
0	-1	1	-1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Tabela 7.13: Quantização dos coeficientes DCT 7.3 -  $F(u,v)/Q(u,v)$ .

Procedendo agora à quantização inversa e à DCT inversa, obtemos respectivamente as tabelas 7.14 e 7.15.

Finalmente colocando novamente os valores no intervalo [0,255] obtemos a tabela 7.16. O erro de final é apresentado na tabela 7.17.

---

Finalmente podemos ver na figura 7.29 a reconstrução de uma imagem com diferentes níveis de quantização. Repare que com passos de quantização pequenos vou ter erros menores, logo melhor qualidade final, mas a compressão vai ser

-80	-44	90	-80	48	40	51	0
-132	-60	-28	0	26	0	0	-55
42	-78	64	0	-120	-57	0	56
0	17	-22	0	51	0	0	0
0	0	-37	0	0	109	0	0
0	-35	55	-64	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Tabela 7.14: Quantização inversa dos coeficientes DCT quantizados 7.5 -  $Q(i,j)$   
 $* (F(u,v)/Q(i,j))$ .

-58	-68	-22	-34	-66	-25	18	48
-43	-27	-43	-53	-26	-1	-35	16
-30	-29	-36	-26	-54	-30	-39	39
4	-75	-17	52	-73	-58	-22	17
45	-71	-14	79	-17	-39	-44	-38
36	-5	3	7	5	-36	-43	34
13	31	41	-55	-22	-27	21	96
22	13	67	-49	-21	19	82	25

Tabela 7.15: DCT inversa dos coeficientes 7.6 -  $\hat{f}(i,j)$ .

70	60	106	94	62	103	146	176
85	101	85	75	102	127	93	144
98	99	92	102	74	98	89	167
132	53	111	180	55	70	106	145
173	57	114	207	111	89	84	90
164	123	131	135	133	92	85	162
141	159	169	73	106	101	149	224
150	141	195	79	107	147	210	153

Tabela 7.16: Valores obtidos depois do processo de compressão/descompressão  
do bloco 7.10 -  $\hat{f}(i,j) + 128$ .

0	10	-6	-24	25	-16	4	11
0	-1	11	4	-15	27	-6	-31
2	-14	24	-23	-4	-11	-3	29
4	16	-24	20	24	1	11	-49
-12	13	-27	-7	-8	-18	12	23
-3	0	16	-2	-20	21	0	-1
5	-12	6	27	-3	2	14	-37
6	5	-6	-9	6	14	-47	44

Tabela 7.17: Erro entre os valores originais 7.10 e os finais 7.16 -  $f(i,j) - \hat{f}(i,j)$ .

inferior. À medida que os passos de quantização vão aumentando a qualidade vai-se degradando e a taxa de compressão vai aumentando.

DCT coding with increasingly coarse quantization, block size 8x8

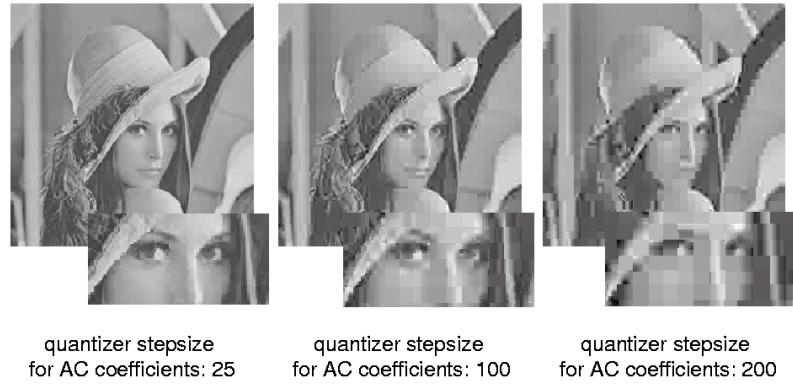


Figura 7.29: Imagem Lena com diferentes níveis de quantização.

### 7.3.6 Codificação entrópica

A etapa de codificação entrópica vai eliminar a informação estatisticamente redundante.

O coeficientes DC e os 63 coeficientes AC são primeiro preparados para a codificação. Eles serão codificados em separado.

O valor DC, como tem o valor mais elevado não será codificado tal e qual, mas codificaremos apenas a sua diferença com o DC do bloco precedente (DPCM Differential Pulse Code Modulation); Os coeficientes AC são linearizados e lidos segundo uma sequência em zigzag como pode ser visto na imagem 7.30.

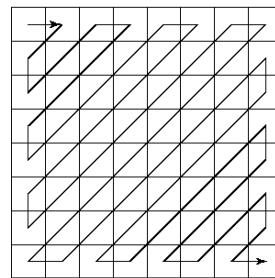


Figura 7.30: Processo de linearização segundo uma sequência em zigzag.

Depois de linearizados é aplicado o método Run length que transforma a sequência em conjuntos do tipo: (# zeros a saltar, próximo valor não zero).

Depois desta preparação aplicamos à sequência obtida uma codificação de Huffman ou codificação aritmética.

O modo de base usa a codificação de Huffman, mas ambas as codificações estão especificadas em todos os modos de operação. A codificação de Huffman

implica que um ou mais conjuntos de tabelas sejam especificadas pela aplicação. As mesmas tabelas são necessárias para descomprimir. Estas tabelas podem ser predefinidas e usadas, por defeito, dentro de uma aplicação, ou calculadas especificamente para uma determinada imagem. As mesmas tabelas serão usadas para codificar todas as amostras de um componente.

No modo sequencial de base apenas existem duas tabelas de Huffman uma para os coeficientes DC e outra para os coeficientes AC.

Vamos de seguida ver estes passos da codificação entrópica separadamente.

#### Codificação entrópica: Primeiro passo (Run Length)

No primeiro passo, cada AC diferente de zero será representado em conjunto com o run-length dos coeficientes AC zero precedentes. A representação é a seguinte:

símbolo-1 (RUNLENGTH, SIZE)	símbolo-2 (AMPLITUDE)
--------------------------------	--------------------------

**RUNLENGTH** número de zeros consecutivos que precedem um AC diferente de zero. Aceita valores entre 0 e 15. Se o RUNLENGTH é maior do que 15 temos o (15,0) que representa 16 zeros consecutivos. Podemos ter no máximo 3 (15,0) seguidos. No caso de termos zeros até ao fim usamos o (0,0) que funciona como *EOB - fim de bloco*.

**SIZE** Número de bits usado para codificar AMPLITUDE usando a codificação de Huffman do JPEG.

**AMPLITUDE** O valor do AC diferente de zero.

Os coeficientes AC são assim representados usando uma sequência de pares *símbolo-1, símbolo-2*, sabendo que cada par pode ter repetições do *símbolo-1* no caso de run-length longos ou pode ter apenas um *símbolo-1* no caso deste ser o (0,0). A relação entre os parâmetros SIZE e AMPLITUDE encontram-se na tabela 7.18.

SIZE	AMPLITUDE
1	-1,1
2	-3,-2,2,3
3	-7,...,-4,4,...,7
4	-15,...,-8,8,...,15
5	-31,...,-16,16,...,31
6	-63,...,-32,32,...,63
7	-127,...,-64,64,...,127
8	-255,...,-128,128,...,255
9	-511,...,-256,256,...,511
A	-1023,...,-512,512,...,1023

Tabela 7.18: Relação entre o valor da amplitude e o número de bits necessários para a codificar (este está representado no parâmetro SIZE).

Quanto aos coeficientes DC a estrutura final é similar aos AC. Neste caso o *símbolo-1* representa apenas o SIZE:

símbolo-1 (SIZE)	símbolo-2 (AMPLITUDE)
---------------------	--------------------------

Neste caso o SIZE pode ir até 11 assim à tabela 7.18 acrescentamos uma linha.

---

**Exemplo** Vamos partir do bloco quantizado apresentado na tabela 7.5 e representá-lo tal como ficaria no final deste passo.

---

(6)(32)(0,3)(6)(0,1)(-1)(0,1)(-1)(1,1)(-1)(3,1)(-1)(2,1)(1)(0,0)

---

### Codificação entrópica: Segundo passo (Codificação de Huffman)

Neste passo primeiro é codificado o coeficiente DC e depois os 63 AC's.

Para AC's e DC o processo é semelhante. Cada *símbolo-1* é codificado usando a tabela de Huffman (VLC's) 7.19 para os DC e 7.20, 7.21 para os AC. Cada *símbolo-2* é codificado usando também códigos de tamanho variável (VLI), mas não são códigos de Huffman.

As tabelas de Huffman terão de ser especificadas externamente.

Diferença dos Coef. DC	SIZE	Código	Tamanho final
0	0	00	2
-1,1	1	010	4
-3,-2,2,3	2	011	5
-7,...,-4,4,...,7	3	100	6
-15,...,-8,8,...,15	4	101	7
-31,...,-16,16,...,31	5	110	8
-63,...,-32,32,...,63	6	1110	10
-127,...,-64,64,...,127	7	11110	12
-255,...,-128,128,...,255	8	111110	14
-511,...,-256,256,...,511	9	1111110	16
-1023,...,-512,512,...,1023	A	11111110	18
-2047,...,-1024,1024,...,2047	B	111111110	20

Tabela 7.19: Tabela de Huffman para os DC.

Note que, o tamanho final em bits corresponde ao tamanho do código mais o SIZE.

### 7.3.7 JPEG: Modo de codificação progressivo

No modo progressivo o passo de DCT e da quantização mantém-se. A principal diferença é que cada componente da imagem é codificado em passagens múltiplas, em vez de uma única passagem. A primeira passagem codifica uma versão aproximada, e as seguintes vão refinando esta.

Existem dois métodos complementares de executar estas passagens. No primeiro apenas um grupo de coeficientes da sequência zigzag são codificados em cada passagem. No segundo método, são codificados primeiro os bits mais significativos de cada coeficiente, e nos passos seguintes vão-se descodificando os outros.

RUN/ SIZE	Tamanho do Código	Código	RUN/ SIZE	Tamanho do Código	Código
0/0 (EOB)	4	1010	5/1	7	1111010
0/1	2	00	5/2	11	11111110111
0/2	2	01	5/3	16	111111110011110
0/3	3	100	5/4	16	111111110011111
0/4	4	1011	5/5	16	1111111110100000
0/5	5	11010	5/6	16	1111111110100001
0/6	7	111000	5/7	16	1111111110100010
0/7	8	1111000	5/8	16	1111111110100011
0/8	10	1111110110	5/9	16	1111111110100100
0/9	16	1111111110000010	5/A	16	1111111110100101
0/A	16	1111111110000011	6/1	7	1111011
1/1	4	1100	6/2	12	1111111101110
1/2	5	11011	6/3	16	1111111110100110
1/3	7	111001	6/4	16	1111111110100111
1/4	9	11110110	6/5	16	1111111110101000
1/5	11	111110110	6/6	16	1111111110101001
1/6	16	11111111100000100	6/7	16	1111111110101010
1/7	16	11111111100000101	6/8	16	1111111110101011
1/8	16	11111111100000110	6/9	16	1111111110101100
1/9	16	11111111100000111	6/A	16	1111111110101101
1/A	16	11111111100001000	7/1	8	11111010
2/1	5	11100	7/2	12	1111111101111
2/2	8	1111001	7/3	16	1111111110101110
2/3	10	111110111	7/4	16	1111111110101111
2/4	12	1111110100	7/5	16	1111111110101000
2/5	16	11111111100001001	7/6	16	11111111101010001
2/6	16	11111111100001010	7/7	16	11111111101010010
2/7	16	11111111100001011	7/8	16	11111111101010011
2/8	16	11111111100001100	7/9	16	11111111101010100
2/9	16	11111111100001101	7/A	16	11111111101010101
2/A	16	11111111100001110	8/1	9	111111000
3/1	6	111010	8/2	15	1111111110000000
3/2	9	11110111	8/3	16	11111111101011010
3/3	12	111111110101	8/4	16	1111111110101111
3/4	16	1111111110001111	8/5	16	1111111110111000
3/5	16	1111111110010000	8/6	16	1111111110111001
3/6	16	1111111110010001	8/7	16	1111111110111010
3/7	16	1111111110010010	8/8	16	1111111110111011
3/8	16	1111111110010011	8/9	16	1111111110111100
3/9	16	1111111110010100	8/A	16	1111111110111101
3/A	16	1111111110010101	9/1	9	111111001
4/1	6	111011	9/2	16	1111111110111110
4/2	10	1111111000	9/3	16	1111111110111111
4/3	16	1111111110010110	9/4	16	1111111111100000
4/4	16	1111111110010111	9/5	16	1111111111100001
4/5	16	1111111110011000	9/6	16	11111111111000010
4/6	16	1111111110011001	9/7	16	11111111111000011
4/7	16	1111111110011010	9/8	16	111111111110000100
4/8	16	1111111110011101	9/9	16	11111111111000101
4/9	16	1111111110011100	9/A	16	11111111111000110

Tabela 7.20: Tabela de Huffman para os AC.

### 7.3.8 JPEG: Modo de codificação sem perdas

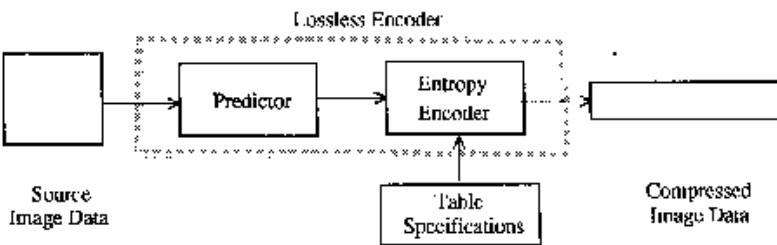


Figura 7.31: Esquema do modo de codificação sem perdas do JPEG.

Este modo aceita amostras com precisão entre 2 e 16. Todas as amostras dos diferentes componentes terão de possuir a mesma precisão.

O preditor combina o valor de no máximo três vizinhos para formar uma previsão de uma amostra. A previsão é subtraída do valor actual e a diferença codificada por um dos codificadores entrópico (Huffman ou aritmético).

RUN/ SIZE	Tamanho do Código	Código	RUN/ SIZE	Tamanho do Código	Código
A/1	9	1111111010	D/1	11	11111111000
A/2	16	1111111111000111	D/2	16	111111111100010
A/3	16	11111111111001000	D/3	16	1111111111100011
A/4	16	11111111111001001	D/4	16	1111111111100100
A/5	16	11111111111001010	D/5	16	1111111111100101
A/6	16	11111111111001011	D/6	16	1111111111100110
A/7	16	11111111111001100	D/7	16	1111111111100111
A/8	16	11111111111001101	D/8	16	1111111111101000
A/9	16	11111111111001110	D/9	16	1111111111101001
A/A	16	11111111111001111	D/A	16	1111111111101010
B/1	10	1111111001	E/1	16	1111111111101011
B/2	16	1111111111010000	E/2	16	1111111111101100
B/3	16	11111111111010001	E/3	16	1111111111101101
B/4	16	11111111111010010	E/4	16	1111111111101110
B/5	16	11111111111010011	E/5	16	1111111111101111
B/6	16	11111111111010100	E/6	16	1111111111100000
B/7	16	11111111111010101	E/7	16	1111111111100001
B/8	16	11111111111010110	E/8	16	1111111111100010
B/9	16	11111111111010111	E/9	16	1111111111100011
B/A	16	11111111111010000	E/A	16	1111111111101100
C/1	10	1111111010	F/0 (ZRL)	11	1111111111001
C/2	16	11111111111010001	F/1	16	111111111110101
C/3	16	1111111111101010	F/2	16	111111111110110
C/4	16	1111111111101011	F/3	16	111111111110111
C/5	16	1111111111101100	F/4	16	1111111111110000
C/6	16	1111111111101101	F/5	16	1111111111110001
C/7	16	1111111111101110	F/6	16	111111111111010
C/8	16	1111111111101111	F/7	16	111111111111011
C/9	16	1111111111100000	F/8	16	111111111111100
C/A	16	1111111111100001	F/9	16	1111111111111101
			F/A	16	1111111111111110

Tabela 7.21: Tabela de Huffman para os AC (continuação).

Qualquer um dos preditores da tabela seguinte podem ser usados excepto o preditor para o valor de selecção 0 que é usados apenas para a codificação diferencial no modo de operação hierárquico.

Valor Selecionado	Predição
0	Sem predição
1	A
2	B
3	C
4	A+B-C
5	A+(B-C)/2
6	B+ (A-C)/2
7	(A+B)/2

Tabela 7.22: Preditores possíveis para o modo de codificação sem perdas do JPEG.

### 7.3.9 JPEG: Modo de codificação hierárquico

O modo de codificação hierárquico permite visualizar a imagem reconstruída em diferentes resoluções. Este modo está representado na figura 7.32.

### 7.3.10 Relação Débito-Qualidade

Apresentamos nas imagens 7.33 e 7.34 alguns resultados do processo de compressão descompressão do JPEG onde pode ser visualizada a relação entre o débito e a qualidade final de uma imagem codificada usando o codificador JPEG.

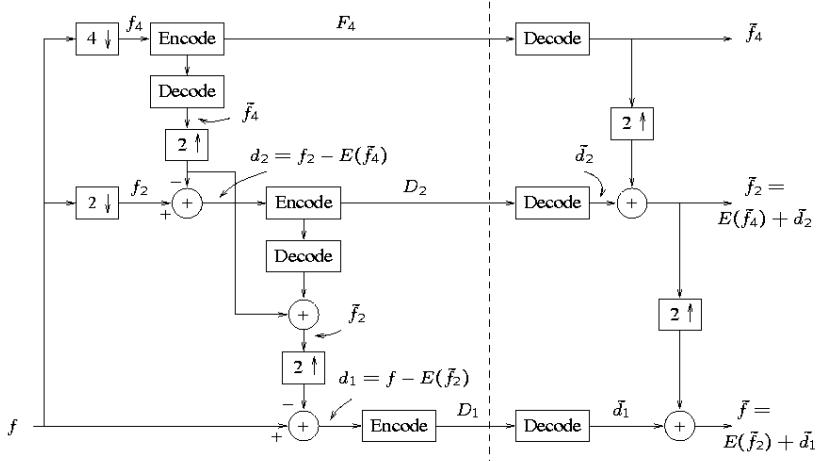


Figura 7.32: Esquema do codificador JPEG hierárquico.



Figura 7.33: Imagem Lena comprimida a diferentes débitos. Cima, esquerda: 65536 Bytes (8 bpp); Cima, direita: 4839 Bytes (0.59 bpp); Baixo, esquerda: 3037 Bytes (0.37 bpp); Baixo, direita: 1818 Bytes (0.22 bpp).

### 7.3.11 MJPEG (Motion JPEG)

MJPEG é uma extensão do JPEG. A ideia base é simples: uma sequência de vídeo digital sendo constituído de imagens sucessivas pode perfeitamente aplicar-se a cada imagem as mesmas técnicas usadas para a imagem fixa.



Figura 7.34: Imagem com cor, original e depois de passar pelo processo de compressão descompressão JPEG.

MJPEG tem vantagens e desvantagens: como trata independentemente cada imagem permite facilidade nos trabalhos de montagem e de pós produção; Mas como ele só não tira proveito das redundâncias temporais, a taxa de compressão permitida para uma boa qualidade de imagem é bastante baixa (2:1 a 5:1, no máximo).

Este tipo de compressão foi adoptado pelos novos formatos de DV (Vídeo Digital).

Algumas características do formato DV e do standard DV são:

- A maior parte dos formatos DV transformam o espaço RGB em  $YC_bC_R$ : 2 : 0(625/50) ou 4 : 1 : 1(525/60);
- O standard DV utiliza a transformada DCT por blocos dois blocos  $4 \times 8$ ;
- A compressão é realizada sobre uma imagem entrelaçada;
- A quantização é continuamente adaptada à complexidade da imagem;
- O sinal vídeo é comprimido grupo a grupo de forma a requerer apenas a leitura de 25 Mbits/s;
- Nestas condições uma hora de vídeo exige um espaço de armazenamento de 11GB que é o proposto pelas mini-cassetes dedicadas ao formato DV.

## Capítulo 8

# Representação de Vídeo Digital

Existem vários tipos de sinais de vídeo. No tipo de vídeo componente são usados três sinais de vídeo separados, um para o vermelho, um para o verde e um para o azul. No tipo de vídeo composto num sinal a crominância e luminância são misturados num único sinal (usa um único fio para transmissão). Já o S-Vídeo - 2 sinais, usa dois sinais, um para a luminância e outro para a composição das duas crominâncias.

### 8.1 Vídeo analógico

No vídeo analógico a amostragem em linhas pode ser feita em modo progressivo ou entrelaçado. Por exemplo, num receptor de televisão, uma imagem é obtida por passagens sucessivas das linhas que constituem o ecrã. Repare que para dar ideia ao olho humano de continuidade teremos de ter um mínimo de 40 projecções por segundo, pois uma frequência entre 20 e 40 produz uma impressão de cintilação. enquanto que abaixo de 20 temos uma sensação de sacada.

Para obter uma sensação de continuidade com 25 imagens por segundo divide-se cada imagem em duas meias imagens projectadas sucessivamente provocando assim 50 sensações diferentes por segundo. Estas meias imagens ou frames são obtidas por passagem primeiro de apenas as linhas pares e depois de apenas as linhas ímpares. A imagem completa é obtida por entrelaçamento das duas tramas.

O número de frames por segundo e a resolução da imagem é determinada pelo standard. Apresentamos de seguida algumas características dos standards PAL (Phase Alternating Line - Europa Ocidental, China e Índia), SECAM (Système Electronique Couleur Avec Mémoire) e NTSC (National Television System Committee - América do Norte e Japão).

**PAL e SECAM** Distinguem 625 linhas das quais apenas 576 são destinadas à imagem, ao resto são afectados serviços de sincronização e de posicionamento dos feixes de electrões (mudanças de linha e de trama). Funcionam em modo entrelaçado com 25 fps (frames por segundo). O modelo de cor

usado é o YUV (Em PAL temos o U e V representados em sinais diferentes e em SECAM temo-los juntos num único sinal);

**NTSC** Este standard afecta apenas 480 das suas 525 linhas para a representação da imagem. Funciona em modo entrelaçado mas com 30 fps (frames por segundo). O modelo de cor usado é o cor em YIQ (um único sinal para I e Q).

A forma de decompor o sinal em tramas e linhas é essencial para a digitalização, pois assim sendo esta apenas terá de se preocupar com a amostragem da linha.

A resolução horizontal não é um parâmetro fixo do sinal de vídeo analógico ela depende da largura da banda disponível ao vídeo e da qualidade do monitor sobre o qual se representará. Por exemplo, sabendo que uma imagem televisiva standard obedece a uma razão de 4:3 (a largura da imagem é 1.33 maior do que a altura), podemos estimar o número de pontos por linha, isto é a largura da imagem em número de pontos.

**Exemplo PAL e SECAM** uma imagem tendo uma altura de 576 linhas, a sua largura será de  $576 \times 4/3 = 760$  pontos.

**NTSC** a largura da imagem será de  $480 \times 4/3 = 640$  pontos.

Em contrapartida, o número total de pontos por linha é de 858 em NTSC e de 864 em PAL e SECAM.

Conhecendo agora o número total de linha por imagem, o número total de pontos por linha e o número total de imagens por segundo (50 para PAL e SECAM e 60 para NTSC), podemos determinar o número total de pontos correspondente a um segundo de vídeo digital UIT-R Rec. 601.

### PAL e SECAM

$$\begin{aligned} 625 \text{ linhas/imagem} \times 864 \text{ pontos/linha} \times 25 \text{ imagens/segundo} &= \\ &= 13500000 \text{ pontos/segundo} \end{aligned}$$

### NTSC

$$\begin{aligned} 525 \text{ linhas/imagem} \times 858 \text{ pontos/linha} \times 30 \text{ imagens/segundo} &= \\ &= 13500000 \text{ pontos/segundo} \end{aligned}$$

Se tivéssemos os valores em RGB teríamos 3 vezes o valor acima, o que representa aproximadamente 40 MB/s. Esta é a razão pela qual mesmo em vídeo analógico escolheu-se separar o sinal digital em luminância e crominância. Para reduzir o débito necessário ao transporte do sinal conservando uma excelente qualidade de imagem graças a uma exploração eficaz das imperfeições da visão humana.

## 8.2 Vídeo digital

O vídeo digital começou por herdar muitas das características do vídeo analógico, mas ele trouxe várias vantagens. O vídeo digital pode ser guardado quer em memória quer em suportes digitais, pronto a ser processado e integrado em várias aplicações multimédia. Com o vídeo digital é possível o acesso directo, o que faz com que a edição não linear seja simples, mais ainda, gravações sucessivas não degradam a qualidade da imagem. O vídeo digital também facilita a encriptação e melhora a tolerância a erros de transmissão.

### 8.2.1 Standard (ITU-R-601)

Em 1970 tomou-se consciência do futuro do vídeo digital e desenvolveu-se um acordo mundial que permitiu em 1982 adoptar um standard único para a televisão digital de qualidade broadcast. Este standard é hoje conhecido por ITU-R-601:

- standard para vídeo componente;
- o número de pontos activos por linha é de 720 para todos os standards;
- usa vídeo entrelaçado;
- foi adoptado pelo SDTV (Standard Definition Television, formato 4:3).

ITU-R-601 reduziu o intervalo de Y de [0, 255] para [16, 235] aplicando a fórmula seguinte:

$$Y' = 219(0.299R' + 0.58G' + 0.114B') + 16$$

A luminância dispõe assim de 220 níveis: o 16 corresponde ao preto e o 235 ao branco. Os restantes valores são utilizados para sincronização e tratamento do sinal.

A UIT-R-601 definiu as expressões seguintes:

$$C_B = 128 + 112 \times \frac{1}{0.886} \times (B' - Y')$$

$$C_R = 128 + 112 \times \frac{1}{0.701} \times (R' - Y')$$

As crominâncias dispõe assim de 225 níveis. Com a diferença de que quando R, G e B são zero estes valores são 128 (o maior possível).

A norma 4:2:2 define o standard da produção enquanto que as normas 4:1:1 e 4:2:0 definem o standard da difusão.

<b>Formato 625/50</b>	<b>Resolução horizontal Luminância</b>	<b>Resolução vertical Luminância</b>	<b>Resolução horizontal Crominância</b>	<b>Resolução vertical Crominância</b>
4:4:4	720	576	720	576
4:2:2	720	576	360	576
4:1:1	720	576	180	576
4:2:0	720	576	360	288

<b>Formato 625/50</b>	<b>Débito útil em Mbit/s (sem compressão)</b>	<b>Débito total em Mbit/s (sem compressão)</b>
4:4:4	248 (8bits) ou 311 (10 bits)	324 (8bits) ou 405 (10 bits)
4:2:2	166 (8 bits) ou 207 (10 bits)	216 (8 bits) ou 270 (10 bits)
4:1:1	124 (8 bits)	162
4:2:0	124 (8 bits)	162

Tabela 8.1: Características dos diferentes formatos 625/50

### 8.2.2 Outros standards

As definições vistas anteriormente tinham em vista aplicações que necessitavam de uma qualidade broadcast (produção e difusão). Outros standards digitais apareceram no final dos anos 80. Realmente era necessário diminuir a largura de banda requerida por aplicações que, como a vídeo conferência, disponham apenas de débitos binários baixos.

Assim foram definidos outros formatos:

- SIF (Source Input Format):

- PAL, SECAM: luminância  $352 \times 288$ ; Crominância  $176 \times 144$ ; 25 imagens/s;
- NTSC: luminância  $352 \times 240$ ; Crominância  $176 \times 120$ ; 30 imagens/s;

- QSIF (Quarter SIF):

- PAL, SECAM: luminância  $176 \times 144$ ; Crominância  $88 \times 72$ ; 25 imagens/s;
- NTSC: luminância  $176 \times 120$ ; Crominância  $88 \times 60$ ; 30 imagens/s;

Ainda para limitar o débito necessário ao transporte do vídeo digital, mas também para permitir a difusão nas duas zonas, de 525 e de 625 linhas, foi definido um formato único:

- CIF (Common Intermediate Format):

- não entrelaçado;
- luminância  $352 \times 288$ ; Crominância  $176 \times 144$ ; 30 imagens/s;
- Para um CIF não comprimido temos um débito binário de  $(352 \times 288 + 176 \times 144 \times 2) \times 30 \times 8 = 36,5 Mbits/s$ . Estes débitos são ainda excessivos para aplicações de vídeo conferência, vídeo sobre PC etc.

- Outros formatos de base CIF:

	Luminância Pixelis por linha	Luminância Linha por imagem	Crominância Pixelis por linha	Crominância Linha por imagem
SQCIF	128	96	64	48
QCIF	176	144	88	72
CIF	352	288	176	144
4CIF	704	576	352	288
16CIF	1408	1152	704	576

Tabela 8.2: Características de formatos com base CIF.

Apesar destes formatos de tamanho reduzido, a integração de vídeo digital num documento multimédia, ou mesmo a difusão deste em qualidade broadcast, são impensáveis sem uma redução drástica dos dados, redução esta que só pode ser obtida por aplicação de algoritmos de compressão eficientes.

## 8.3 Compressão de vídeo

Na compressão de vídeo vamos tirar proveito das redundâncias:

- espaciais;
- psicosensoriais;
- estatísticas;
- temporais.

### 8.3.1 Compressão temporal

Temos compressão temporal sempre que exploramos as semelhanças existentes entre imagens sucessivas. Trata-se de identificar as informações redundantes no tempo, mesmo se elas mudaram de lugar no espaço. Estas semelhanças podem ser constatadas entre: a imagem corrente e a imagem precedente - análise unidireccional; ou entre a imagem corrente e as duas imagens que a englobam (a precedente e a seguinte) - análise bidireccional.

#### Compressão temporal unidireccional

Depois da primeira imagem de uma sequência podemos codificar apenas as diferenças que distinguem uma dada imagem da imagem anterior.

Facilmente verificamos que este processo não poderá ser efectuado para a primeira imagem da sequência. Esta terá de ser codificada apenas explorando as redundâncias espaciais existentes no interior da imagem.

As imagens que são codificadas apenas espacialmente chamamos imagem *intra* ou *key frame*. Às imagem obtida da diferença de uma imagem com a precedente chamamos imagem *delta*. As imagens *delta* podem de seguida ser submetidas a uma compressão espacial.

Para comprimir uma sequência desta forma temos de dispor simultaneamente de duas imagens sucessivas (a imagem  $n$  e a imagem  $n - 1$  usada para o cálculo das diferenças). Para descomprimir a sequência o descompressor vai primeiro proceder à descompressão espacial antes de juntar as diferenças à imagem precedente para obter a imagem corrente.

Poderíamos imaginar que uma sequência de vídeo fosse comprimida segundo esta técnica simples: a primeira imagem *intra* seguida de todas as outras *deltas*. Uma tal sequência apresentaria graves inconvenientes:

- para representar uma imagem qualquer teríamos de partir sempre da primeira imagem *intra* para de seguida calcular todas as outras imagens sucessivas até chegar à imagem que nos interessava. Assim, seria impossível começar a leitura de um vídeo a meio de uma sequência, coisa que fazemos muito frequentemente, quando por exemplo, ao ver televisão mudamos de canal.
- Outro inconveniente seria que o menor erro provocado pela transmissão ou pela descodificação seria propagado por todas as imagens seguintes.
- Além destes inconvenientes, a diferença entre duas imagens sucessivas pode ser tal que é preferível guardar a imagem actual como *intra*. É por exemplo o caso de mudança de cena.

Os inconvenientes precedentes levam-nos a concluir que um procedimento apenas diferencial tornaria impossível qualquer montagem virtual. Consequentemente, é obrigatória a introdução frequente de imagens *intra*.

A frequência de imagens *intra* é determinada pelo dinamismo da sequência. Assim, uma sequência muito animada deverá incluir muito mais imagens *intra* que uma entrevista apresentando uma pessoa estática sobre um fundo fixo.

Esta técnica é utilizada pelos algoritmos de compressão do tipo:

- Cinepak - desenvolvida pela *Compression Technologies, Inc* e, na sua origem, destinada à compressão de vídeo para os CDROMs de velocidade simples.
- Indeo (Intel Video) - criada pela Intel (Indeo - Intel Video) e foi principalmente conhecida pelo nome Real Time Video 2.1 (RT21).

### **Compressão temporal bidireccional**

Um movimento lateral da câmara faz aparecer sobre a imagem elementos novos sobre os quais as imagens anteriores não continham qualquer informação. Neste caso, se procedemos a um cálculo diferencial a partir da imagem  $n-1$ , o tamanho da imagem *delta* correspondente à imagem  $n$  é frequentemente grande, pois apenas alguns dos elementos são comuns às duas imagens. Mas se compararmos a imagem  $n$  não apenas com  $n-1$  mas também com  $n+1$ , é possível aplicar uma codificação diferencial eficaz à imagem  $n$  na qual uma parte de elementos é comum com  $n-1$  e outra parte é comum com  $n+1$ .

### **Compensação de movimento**

Notemos que o movimento dos elementos de uma imagem à seguinte aumenta consideravelmente as diferenças. Temos de encontrar uma solução susceptível de limitar o volume das informações diferenciais a codificar.

A técnica usada para minimizar este problema denomina-se *compensação de movimento*. Não vamos calcular as diferenças de um bloco com o mesmo bloco na imagem anterior, ou seguinte, mas com os blocos que mais se aproximam do bloco que estou a tratar. Neste caso terei também de guardar os movimentos em forma de vectores (vectores de movimento).

A compressão bidireccional e a compensação de movimento são muito utilizadas pelos algoritmos de compressão de vídeo MPEG.

### **8.3.2 Norma MPEG**

MPEG é o diminutivo de "Moving Picture Experts Group". Este grupo criou a norma de vídeo e áudio que vai permitir entre 1989 e 1992 um grande desenvolvimento à difusão do vídeo digital. Esta norma foi adoptada em fins de 1992 sob a referência ISO 11172 e baptizada com o nome do grupo que a desenvolveu: MPEG.

Em 1989 a IUT-T define a norma H.261, que já utiliza técnicas híbridas de compressão intra trama (explorando a redundância espacial) e de compressão inter trama (explorando a redundância temporal, graças às primeiras técnicas elaboradas para estimar a compensação do movimento). Concebida para aplicações que apenas dispunham de baixos débitos binários ( $p \times 64$  Kbits/s,  $p$  podendo tomar valores entre 1 e 30). Ela prevê a codificação de vídeo por componentes  $YCB_C_R$ . Em contrapartida, o formato é limitado ao formato QCIF, sendo os formatos SQCIF e CIF acessíveis como opção. Dados os fracos débitos binários autorizados, a qualidade de vídeo sofre fortes degradações desde que apareçam movimentos rápidos ou conteúdos não estáveis no tempo. Estas limitações confinam o H.261 a aplicações do tipo vídeo-fonia ou vídeo-conferencia. No entanto este standard é o predecessor do MPEG.

MPEG pertence à classe geral de algoritmos de compressão híbridos do tipo predição-transformação. O termo híbrido significa que várias técnicas são empregues conjuntamente para aumentar a eficiência global do sistema:

- a predição é aplicada para a compressão temporal;
- a transformada DCT seguida da quantização dos coeficientes transformados assegura uma codificação espacial;
- ao fim, uma codificação entrópica optimiza o código final.

A compressão MPEG é assimétrica: a codificação é muito mais complexa (4 vezes mais longa) do que a descodificação.

### Especificações MPEG

- MPEG-1 Combinação de áudio e vídeo digital num único ficheiro, para armazenamento ou transmissão. Atinge débitos da ordem dos 1.5 Mbits/s (CD).
- MPEG-2 melhora a qualidade do vídeo e assegura a adaptação a débitos binários maiores (de 6 a 40 Mbits/s) e a maior número de formatos (4 vezes mais do que MPEG-1). Com uma representação de 50 tramas/s, MPEG-2 é destinado ao vídeo de qualidade broadcast e às aplicações HDTV (débitos binários de 20 a 40 Mbits/s).

Uma norma MPEG-3 deveria ser elaborada para o HDTV. No entanto, a partir de 1992-1993, as especificações HDTV foram integradas ao MPEG-2, sendo a qualidade final deste suficiente para tornar inútil o nascimento do MPEG-3.

- MPEG-4 propõe uma norma de compressão satisfazendo as aplicações que dispõe de débitos muito baixos. Os campos que pretende cobrir são: televisão digital; aplicações gráficas interactivas, multimédia interactiva (WWW, distribuição e acesso a conteúdos)
- MPEG-4 Parte 10 ou H264 - Codificação avançada de vídeo;
- MPEG-7 - Interface para descrição de conteúdos multimédia;
- MPEG-21 - "Conteúdos a qualquer hora e em qualquer lugar".

#### 8.3.3 MPEG-1

O objectivo do "Moving Picture Experts Group" é muito ambicioso pois trata-se de permitir a restituição, a partir de um CD de 72 minutos de vídeo digital no ecrã todo (imagem e som) ao ritmo de 25 ou 30 imagens por segundo.

A primeira comercialização de produtos correspondentes a esta norma foi realizada pela Philips (CDI - Compact Disc Interactive) no início de 1994: consegui-se passar o vídeo digital para 1.44 Mbits/s, para a imagem e o som.

Relembremos no entanto que é necessário um débito binário de 166 Mbits/s para a restituição, sem o som, de vídeo digital 4:2:2. Isto contra o 1.2 Mbits/s conseguido pela MPEG-1 implica uma taxa de compressão de aproximadamente 140:1. Registamos no disco de suporte menos de 1% dos dados originais do vídeo.

A qualidade proposta não é a da difusão broadcast, mas ela é comparável à obtida a partir do VHS.

Para conseguir baixos débitos a norma MPEG1 toma em conta todos os métodos de compressão:

- Em primeiro usa a redução de formato usando o formato SIF que como já vimos propõe  $352 \times 288$  pixéis de luminância e  $176 \times 144$  pixéis em crominância.
- Depois só toma uma imagem sobre duas do vídeo original (usando a representação entrelaçada). Para diminuir ainda o débito binário usa a norma 4:2:0.
- Quanto à compressão espacial ela funciona tal como já funcionava o MJPEG.
- É no tratamento das redundâncias temporais que MPEG propõe soluções novas.

### Tratamento das redundâncias temporais

MPEG-1 define três tipos diferentes de imagem que serão tratadas usando procedimentos específicos:

- As imagens *intra* ou imagens de referência são simbolizadas pela letra **I**;
- As imagens preditas ou imagens **P**;
- As imagens bidirecionais ou imagens **B**.

Vamos de seguida ver como são tratadas cada uma destas imagens.

**Imagens I** Estas imagens são comprimidas segundo um algoritmo JPEG, elas servem de referência para o cálculo das imagens **P** e **B** e para a decodificação. É por isso que uma imagem em cada 12, ou menos deverá ser uma imagem intra.

**Imagens preditas ou tipo P** Estas imagens são constituídas a partir da imagem **I** ou **P** precedente, usando vectores de movimento e cálculo de diferenças.

Para diminuir o erro de predição, uma compensação de movimento é aplicada aos macro-blocos (dois blocos de crominância de  $8 \times 8$  pixéis e 4 blocos de luminância de  $8 \times 8$  pixéis ou seja um bloco de  $16 \times 16$  pixéis); Se o macro bloco é encontrado, ele é codificado como a sua diferença com os valores na imagem precedente e depois quantizado; finalmente, aplica-se uma codificação estatística aos valores obtidos depois da quantização. Se o macro-bloco não é encontrado, ele será codificado seguindo a norma JPEG como se se tratasse de uma imagem **I**;

Os vectores de movimento são calculados segundo o deslocamento de cada bloco de pixéis dumha imagem à outra, e depois codificados;

Evidentemente, e contrariamente às imagens **I** que são descodificadas como imagens JPEG, a decodificação das imagens **P** necessita que o decodificador guarde a imagem precedente. A decodificação da imagem actual

é efectuada a partir dos macro-blocos codificados em directo e os macro-blocos codificados pelas diferenças.

Na compensação de movimento a norma MPEG não especifica nem como é feita a procura, nem até que ponto a fazer, nem mesmo limite de proximidade a esperar. Estes parâmetros deverão ser definidos por cada implementação do algoritmo. Por exemplo, uma implementação particular pode procurar a mesma posição na imagem precedente e depois em todas as direcções possíveis deslocando-se de um valor  $dx$  em  $X$  e de  $dy$  em  $Y$ .

**Imagens bidireccionais ou do tipo B** Estas imagens são construídas a partir das imagens **I** ou **P** precedentes e seguintes. Neste caso teremos vectores de movimento para a frente e para trás. Cada bloco de pixeis de uma imagem **B** toma como valor a média entre o mesmo bloco na imagem precedente e na imagem seguinte.

Para descodificar este tipo de imagens o descodificador terá de ter simultaneamente três imagens em memória. A **I** ou **P** precedentes, a **I** ou **P** seguintes e a **B** actual.

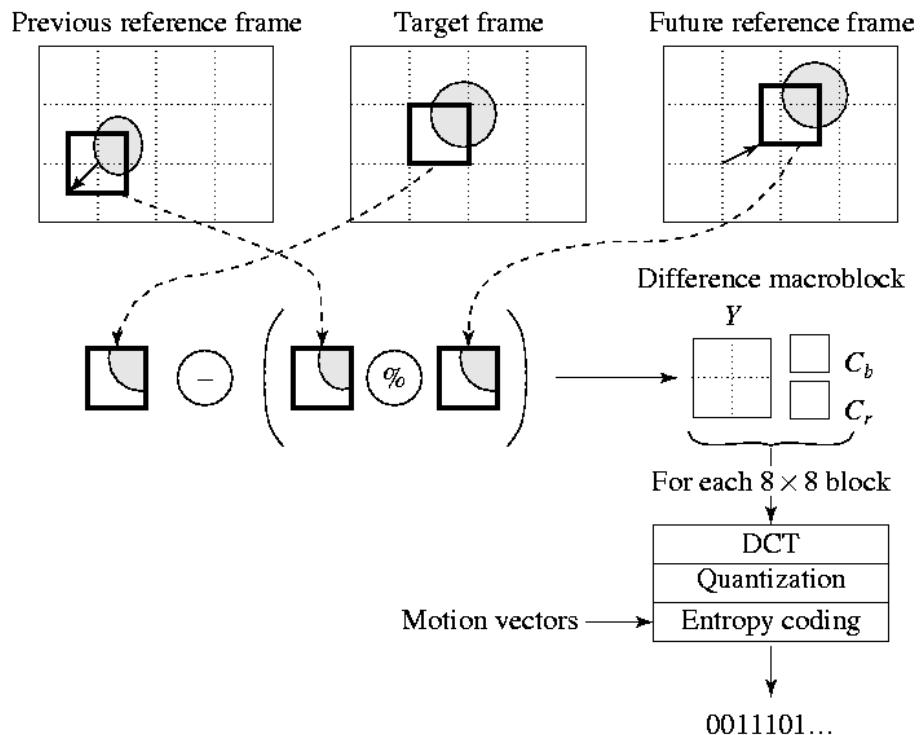


Figura 8.1: Calculo de uma frame B.

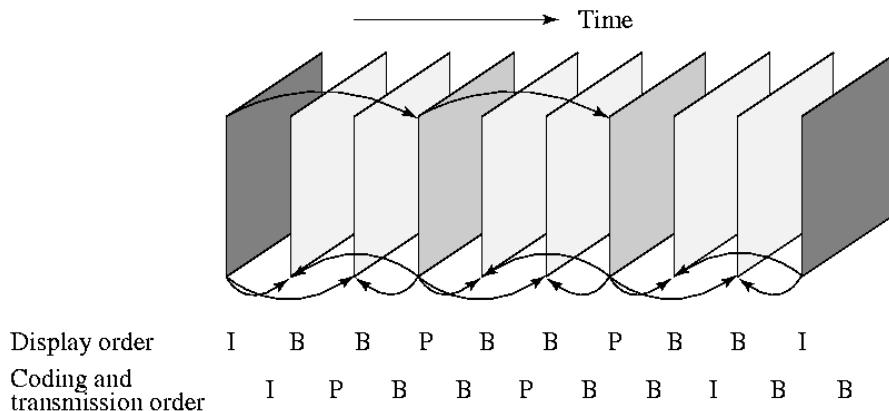


Figura 8.2: Organização das frames no MPEG.

Tipos de imagem	Tamanho (em Ko)	Compressão
Intra ( <b>I</b> )	18	7:1
Preditas ( <b>P</b> )	6	20:1
Bidirecionais ( <b>B</b> )	2.5	50:1
Média	4.8	27:1

Tabela 8.3: Tipos de imagens e taxas de compressão

### O som na norma MPEG-1

Uma sequência de vídeo integra necessariamente uma banda de som. O MPEG-1 trata este média com o mesmo cuidado que a imagem.

O algoritmo de compressão de áudio usado no MPEG-1 é fundado sobre standard MUSICAM (Masking-pattern Universal Sub-band Integrated Coding and Multiplexing). Ele permite obter um sinal estéreo com amostragem 48 kHz e quantificado sobre 16 bits (ou seja um débito de  $2 \times 768$  Kbits/s) e permite quatro taxas de compressão. O MPEG-1 utiliza a taxa 8 : 1 para obter uma qualidade final do tipo Hi-Fi.

O princípio geral desta compressão áudio é relativamente simples. Suprimimos do sinal sonoro as redundâncias psicosensoriais, o que quer dizer, tudo o que o ouvido humano não percebe ou percebe mal. Para começar eliminam-se todos os elementos cuja amplitude está situada abaixo do limiar de sensibilidade do ouvido humano. Elimina-se de seguida todas as frequências encobertas por outra frequência vizinha de amplitude mais forte. Finalmente, efectua-se uma quantização mais fina aos sinais com frequências para as quais o ouvido é mais sensível do que para os sinais para os quais o ouvido é menos sensível.

### Sincronização

Funcionando os codecs de áudio e de vídeo de forma completamente independente no tratamento dos seus dados específicos, será necessário sincronizar o fluxo de áudio e de vídeo. É por isso que o MPEG-1 é constituído por três

partes: áudio, vídeo e sistema. Este último assegura a integração dos dois primeiros.

O MPEG-1 dispõe de um relógio de sistema a 90 Mhz que indica a hora aos codificadores áudio e vídeo. Os marcadores temporais gerados são incluídos no código e enviados ao destinatário que os utiliza para sincronizar o som e a imagem.

### 8.3.4 MPEG-2

O MPEG-2 é uma extensão de MPEG-1 desenvolvido com a finalidade de obter uma qualidade superior de imagem e som.

O objectivo desta vez foi conseguir uma qualidade equivalente ou superior à do vídeo composto com débito na ordem dos 4 a 8 Mbits/s e para imagens de alta qualidade, com débitos de 10 a 15 Mbits/s.

Os domínios de aplicação visados são a difusão de canais de televisão digital por cabo ou satélite, assim como a distribuição de alta qualidade sobre DVD (Digital Versatile Disc).

Os procedimentos de compressão espaciais e temporais usados por MPEG-2 são directamente tomados do MPEG-1. A diferença essencial entre estes dois standards consiste na definição própria ao MPEG-2 de vários níveis de qualidade e perfis de codificação específicos.

#### Os níveis de MPEG-2

Quatro níveis definem a resolução da imagem em MPEG-2:

1. Nível baixo: Resolução  $352 \times 288$ ;
2. Nível médio: Resolução  $720 \times 576$ ;
3. Nível alto-1440: Resolução  $1440 \times 1152$ ;
4. Nível alto: Resolução  $1920 \times 1152$ .

#### Os perfis de MPEG-2

MPEG-2 propõe 5 perfis diferentes, cada um oferecendo um conjunto de especificações de técnicas de compressão:

1. Perfil simples: dispõe apenas de imagens **I** e **P**;
2. Perfil médio: esquema herdado de MPEG-1;
3. Perfil escalável em SNR: permite uma recepção sobre dois níveis graças à organização de duas tramas distintas de dados. Permite assim que se descodifique apenas a primeira trama. A segunda vem melhorar a relação sinal/ruído.
4. Perfil escalável espacialmente: Idêntico ao anterior, mas dispondo de uma trama de dados suplementares permitindo obter uma imagem em alta definição a partir de um formato TV  $720 \times 576$ .
5. Perfil alto: dispõe de todas as técnicas dos perfis anteriores, mas codifica o sinal vídeo 4:2:2 (em vez de 4:2:0 como nos outros perfis).

Com os quatro níveis e cinco perfis anteriores o MPEG-2 permite 11 combinações. Na tabela abaixo apresentamos os débitos máximos (Mbits/s) para diferentes níveis de perfis na norma MPEG-2.

Perfis Níveis	Simples	Médio	Escalável em SNR	Escalável espacialmente	Alto
<b>Alto</b>	Não retido	80	Não retido	Não retido	100
<b>Alto-1440</b>	Não retido	60	Não retido	60	80
<b>Médio</b>	15	15	15	Não retido	20
<b>Baixo</b>	Não retido	4	4	Não retido	Não retido

Tabela 8.4: Combinação dos níveis e perfis no MPEG-2

### 8.3.5 MPEG-4

O MPEG-4 é muito mais do que uma evolução de MPEG-1 e MPEG-2. Ele segue uma abordagem nova da codificação e da compressão de imagem animada e do som, fundada sobre uma análise do conteúdo em objectos (visuais ou sonoros) estruturados hierarquicamente e susceptíveis de tratamentos específicos.

No MPEG-4, além da compressão (agora o débito binário pode ir de 5 kbps a 10 Mbps), também dá grande importância à interacção com o utilizador. Este permite uma melhor gestão e protecção dos direitos de autor.

Alguns exemplos de implementações são: RelPlayer, QuickTime, GPAC/Osmo4, XviD, MPEG4IP, iTunes, VLC, 3ivx, Nero Digital, etc.

O MPEG-4 fornece diversas formas estandardizadas:

- Objectos de média: representação de unidades de conteúdo áudio, vídeo, ou áudio-visual. Estes objectos podem ser naturais ou sintetizados, isto é, vindos de uma câmara ou um micro, ou realizados num computador: gráficos 2D ou 3D, sons sintetizados;
- Objectos de média compostos: descrição da composição dos objectos na criação dos objectos compostos que formam as cenas áudio-visuais;
- Descrição da cena: define, as relações espaço-temporais entre os objectos.
- Interacção com a cena áudio-visual gerada no receptor;
- Identificação de propriedade intelectual.

#### Decomposição da cena em objectos

O MPEG-4 propõe uma abordagem orientada ao objecto: uma cena de vídeo é decomposta em objectos que podem ser:

- Imagens fixas, por exemplo um fundo fixo;
- Objectos de vídeo, por exemplo uma pessoa que fala, sem o fundo;

- Objectos áudio, por exemplo a voz associada com a pessoa, ou música de fundo;
- Texto e gráficos;
- Sons sintetizados;
- rostos que falam, sintetizados, associados a texto usado para sintetizar um discurso e animar o rosto; corpos animados para juntar aos rostos;

Estes objectos podem ainda ser decompostos e depois estruturados e hierarquizados em árvore. Nas folhas destas árvores temos os objectos simples.

É importante notar que na sua forma codificada, cada objecto é representado independentemente dos objectos que o rodeiam ou do fundo. É claro que a codificação específica dos objectos ou a integração dos objectos sintetizados numa sequência de vídeo supõe que sejamos capazes de separar os objectos do fundo sobre o qual estes aparecem. Este tipo de decomposição permite que os vídeos em MPEG-4 possam ser compostos e manipulados com operações simples sobre os objectos visuais.

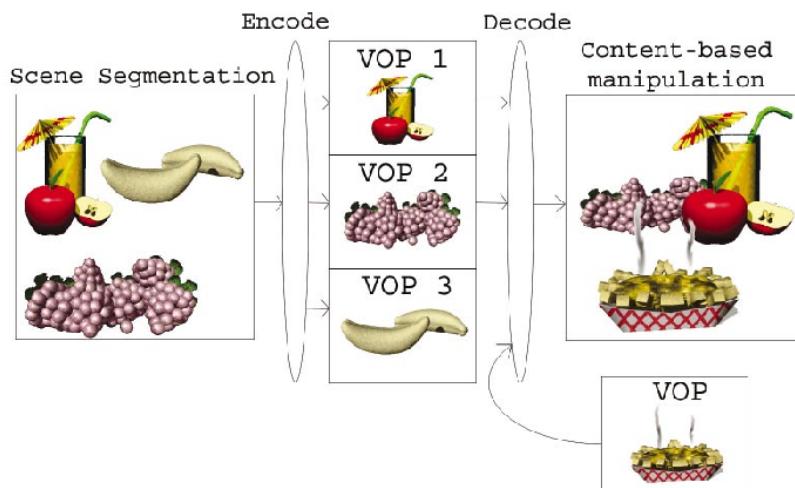


Figura 8.3: Segmentação de uma cena em objectos.

### Composição de objectos

Objectos compostos agrupam vários objectos simples. Objectos simples correspondem a folhas numa árvore descriptiva enquanto que objectos compostos englobam sub-árvores inteiras. Este agrupamento permite aos autores construir cenas complexas e permite aos consumidores manipular conjuntos de objectos (permite por exemplo manipular uma pessoa e a sua voz simultaneamente).

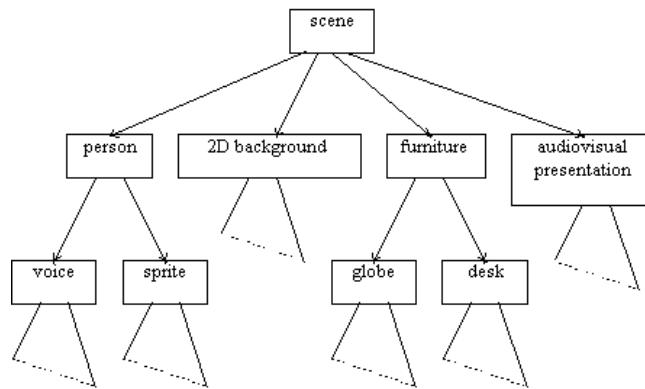


Figura 8.4: Decomposição dos objectos.

### Descrição da cena

O MPEG-4 fornece uma estandardização da descrição da cena. Esta permite por exemplo:

- colocar objectos em qualquer lado num determinado sistema de coordenadas;
- aplicar transformadas para alterar a aparência geométrica ou acústica de um objecto;
- agrupar objectos simples para formar objectos compostos;
- modificar atributos dos objectos;
- alterar interactivamente, os pontos de visualização e audição para qualquer ponto da cena.

Para permitir relacionar tramas binárias elementares com objectos dentro de uma cena usamos os descritores de objectos.

Um descritor de um objecto deve identificar todos os dados binários associados a um objecto, podendo estes estar distribuídos na trama binária. Isto permite manusear hierarquicamente os dados codificados e também a meta-informação associada ao conteúdo e direitos sobre a propriedade intelectual associada a estes.

Como se trata de uma cena animada à descrição do conteúdo da cena teremos de juntar a descrição da evolução deste conteúdo no tempo. Assim, esta análise estruturada da cena em objectos deve integrar uma sincronização espacial (posição dos objectos na cena) e uma sincronização temporal absoluta (cada objecto relativamente à cena) e relativa (entre objectos).

A descrição da cena assenta sobre conceitos de realidade virtual (VRML) quer em termos de estrutura quer em termos de funcionalidade.

A descrição da cena pode ser realizada em dois passos:

- Num primeiro passo, sobre forma textual (podemos utilizar uma descrição estandardizada chamada XMT- eXtensible MPEG-4 Textual format- ou uma forma mais pessoal contentando-se de usar a sintaxe XML, VRML

ou outra) define-se a relação entre as tramas binárias elementares, pertencendo a cada objecto.

- Num segundo passo, descreve-se a organização espaço-temporal dos objectos na cena em formato binário BIFS - "Binary Format for Scenes".

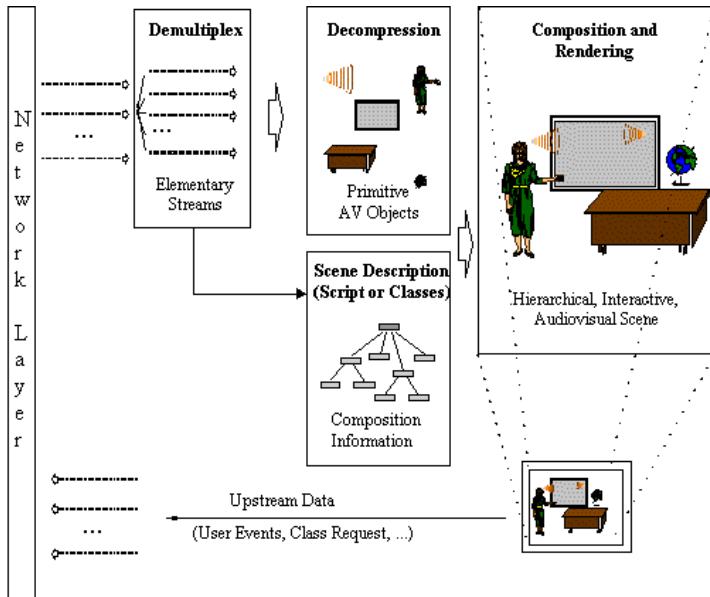


Figura 8.5: Descrição da cena.

### Adaptação dimensional

A exploração desta descrição detalhada da cena não é usada unicamente para a optimização da compressão objecto a objecto. Esta nova abordagem permite igualmente a aplicação de tratamentos específicos adaptados a cada objecto, permitindo assim novas funcionalidade essenciais à difusão e utilização de aplicações multimédia: adaptação dimensional e interacção.

A adaptação dimensional (scalability) permite optimizar a difusão do fluxo de dados em função do débito e/ou do sistema de representação que possui o utilizador final. O autor da sequência codifica cada objecto sobre vários níveis (layers) correspondendo a diferentes graus de qualidade e complexidade. A adaptação diz respeito aos parâmetros qualitativos de ordem temporal (número de imagens por segundo), espacial (resolução da imagem), ou ainda relativos ao nível de perda ligada a compressão (SNR). Assim, a reconstituição das imagens e dos sons da sequência será perfeitamente adaptado às condições locais de utilização (ligação de alto ou baixo débito, recepção sobre um televisor, um PDA, ou um telemóvel, etc.).

### Interacção com os objectos

Em geral o utilizador observa uma cena que é composta seguindo o desenho do autor da cena. Dependendo do grau de liberdade fornecida pelo autor, o

utilizador tem possibilidades de interacção com a cena. Em particular pode alterar atributos da descrição da cena. Algumas operações que o utilizador poderá efectuar incluem:

- Mudar o ponto de visualização ou audição da cena;
- Mudar objectos da cena para outros locais;
- alterar a cor de um objecto apresentado sobre um site de comércio eletrónico;
- Alterar o tamanho de texto sintetizado;
- Despoletar um conjunto de acontecimentos, clicando num objecto específico, por exemplo começar ou terminar uma sequência de vídeo;
- Escolher uma determinada linguagem quando várias estão disponíveis;
- etc.

#### Codificação baseada nos objectos vs. codificação baseada em blocos

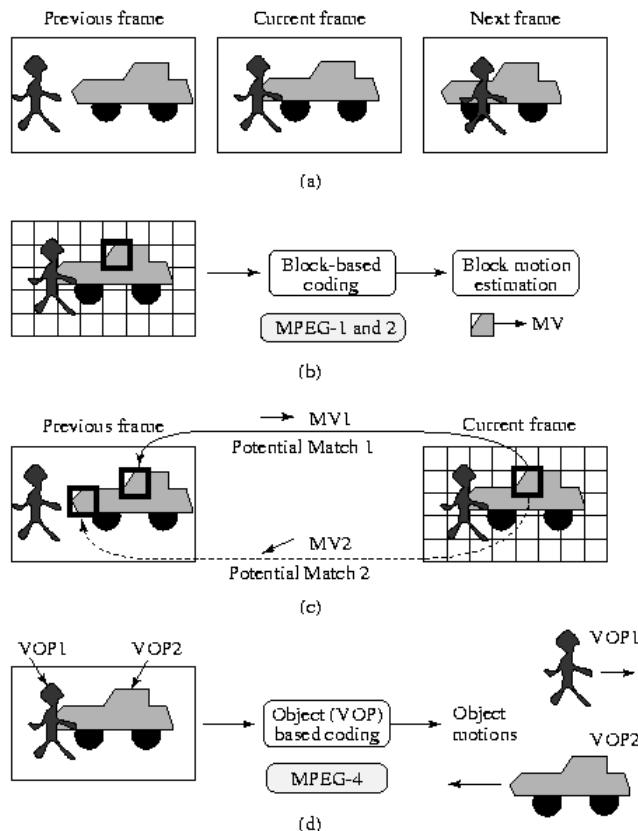


Figura 8.6: Codificação baseada nos objectos vs. codificação baseada em blocos.

### Codificação baseada nos objectos de vídeo (VOP)

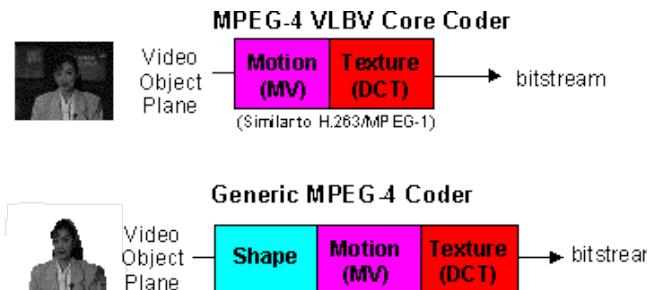


Figura 8.7: Codificação baseada nos objectos.

A codificação de imagens e vídeos convencionais é similar à apresentada para o MPEG-1/2. Envolve estimação do movimento, predição compensada em movimento seguida de codificação de texturas.

Para as funcionalidades baseadas no conteúdo, onde a sequência de imagens de entrada pode ter uma forma e localização arbitrária, a codificação anterior é estendida pela codificação da informação referente à forma e à transparência.

A codificação baseada em conteúdos pode melhorar significativamente a eficiência da compressão. Algumas técnicas específicas da codificação baseada em conteúdos são:

- Compensação de movimento global para objectos de vídeo;
- Compensação global de movimento baseada em "sprites" estáticos: um "sprite" estático é uma imagem estática, possivelmente grande, que descreve o fundo panorâmico de várias frames da sequência. Para cada imagem consecutiva dumha sequência, apenas os parâmetros globais de movimento, descrevendo o movimento da câmara, são codificados para reconstruir o objecto.

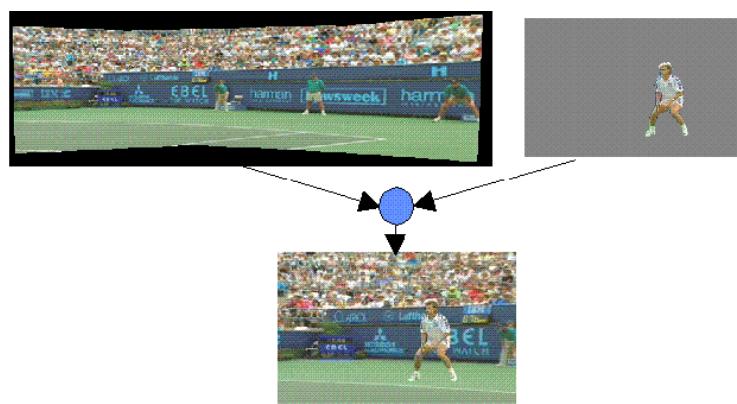


Figura 8.8: Técnica de "sprites".

- DCT adaptada à forma SA-DCT.

### 8.3.6 Codificação das texturas dos VOP

A codificação das texturas dos VOP pode usar a DCT ou a SA-DCT (Shape Adaptive DCT). Na DCT as partes dos macro-blocos fronteira fora do VOP são preenchidas com zeros antes de serem enviados para a DCT. A SA-DCT é uma 2D DCT e é calculada como uma transformada 2D separável em duas iterações da 1D DCT-N. A transformada 1D DCT-N e a sua inversa, 1D IDCT-N são definidas respectivamente nas equações (8.1) e (8.2), onde  $i = 0, 1, \dots, N-1$ ,  $u = 0, 1, \dots, N-1$ .

$$F(u) = \sqrt{\frac{2}{N}} C(u) \sum_{i=0}^{N-1} \cos \frac{(2i+1)u\pi}{2N} f(i). \quad (8.1)$$

$$\tilde{f}(u) = \sum_{i=0}^{N-1} \sqrt{\frac{2}{N}} C(u) \cos \frac{(2i+1)u\pi}{2N} F(u). \quad (8.2)$$

$$C(u) = \begin{cases} \frac{\sqrt{2}}{2} & \text{se } u = 0, \\ 1 & \text{noutro caso.} \end{cases} \quad (8.3)$$

#### Codificação de objectos sintetizados

O MPEG-4 permite a codificação de objectos sintetizados, nomeadamente: animação de rostos; animação de corpos; malhas 2D animadas; malhas 3D, etc.

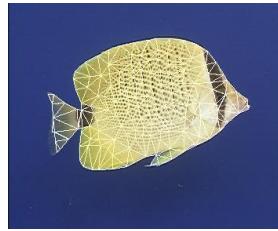


Figura 8.9: Objecto sintetizados.

#### Perfis do MPEG-4

O MPEG-4 define, vários perfis adaptados a vários tipos de aplicação. Estes perfis constituem subconjuntos das técnicas de vídeo, áudio e sistema do MPEG-4, permitindo limitar ao estritamente necessário o número de técnicas que um codificador deverá implementar.

O MPEG-4 propõe os 5 seguintes tipos de perfis:

- perfil visual: com cinco níveis de exigência para a codificação de dados naturais e quatro níveis para a codificação de dados sintetizados ou mistos (naturais / sintetizados);
- perfil áudio: com quatro níveis definidos pelo número de técnicas integradas e o débito requerido;

- perfis gráficos: com três níveis de tratamento de elementos gráficos e textuais;
- perfis de descrição da cena: neste os quatro níveis determinam os tipos de informação susceptíveis de ser integradas na descrição de uma cena (desde apenas informações áudio até ao 3D);
- perfis de descrição de objectos: dispõe de um nível único e propondo as técnicas seguintes: descritor de objectos; sincronização; informação sobre os objectos, propriedade intelectual e protecção.

### 8.3.7 H.264 - Codificação avançada de vídeo

Depois da norma H.261 o grupo VCEG (Video Coding Experts Group) da UIT-T propôs ainda as normas H.263 (1995) e H.263+ (1998). Em 1998 iniciou o projecto H.26L cujo objectivo é obter uma eficiência de codificação que seja o dobro das obtidas pelos standards existentes.

Em Dezembro de 2001 VCEG e MPEG formam o JVT (Joint Video Team) cuja missão é de finalizar um novo standard de codificação: H.264 ou AVC (Advanced Vide Coding), igualmente designado por MPEG-4 parte 10.

H.264/AVC é fiel ao espírito dos outros standards de compressão vídeo. Os elementos funcionais de base (predição, transformação, quantização, codificação entrópica) são um pouco diferentes dos standards precedentes; as alterações mais importantes dizem respeito aos detalhes funcionais de cada um dos elementos funcionais.

Em relação ao MPEG-2, o H.264/AVC é entre 2 a 3 vezes mais complexo em decodificação e 4 a 5 vezes mais complexo em codificação. Esta complexidade permite um aumento de eficiência que permite reduzir 39 % o débito binário relativamente ao MPEG-4 e 65% relativamente ao MPEG-2.

#### Características gerais do standard H.264/AVC

O standard H.264/AVC aceita a codificação de vídeo no formato 4:2:0, progressivo ou entrelaçado, podendo estes dois modos de representação ser combinados numa mesma sequência; Como nas outras normas de compressão de vídeo, cada macro-bloco é transformado, quantizado e codificado.

Algumas das melhorias introduzidas são:

- A estimativa de movimento é enriquecido: esta é realizada no interior do macro-bloco, cuja resolução pode variar de  $16 \times 16$  pixéis a  $4 \times 4$  pixéis, passando por várias combinações intermédias.
- A escolha do tamanho dos sub-blocos tem um grande impacto na eficiência da compressão e sobre o tempo de cálculo. Em geral, uma partição mínima (blocos de  $16 \times 16$ ) deve ser preferida no caso de zonas homogéneas da imagem e uma partição máxima (blocos de  $4 \times 4$ ) pode ser vantajoso para zonas contendo numerosos detalhes.
- A transformação e a quantização operam sobre blocos de tamanho variável e não apenas blocos de  $8 \times 8$  pixéis;

- A transformação é baseada na DCT, mas com algumas diferenças importantes: algumas astúcias matemáticas permitem que a transformação se resuma a uma multiplicação matricial;
- Usa 52 quantizadores escalares, não lineares (a série começa em 0.625 e termina em 224 com um crescimento de 12.5% de um valor a outro);
- A codificação entrópica é enriquecida (CABAC - Context Adaptive Binary Arithmetic Coding);
- É usado um filtro na fase de descompressão para melhorar a qualidade visual subjetiva por redução do efeito de mosaico.

### 8.3.8 Os formatos de ficheiro de vídeo digital

#### Formatos de produção de vídeo

Os formatos de produção correspondem aos diferentes tipos de câmaras ou visualizadores/gravadores digitais, usados para a visualização ou gravação de sequências de vídeo.

Formatos	Amostragem	Taxa de compressão	Algoritmos de compressão	Resolução da luminância	Débito em Mbits/s
DV	4:2:0	5:1	DV Intra-image	720 × 576	25
	4:1:1			720 × 480	
DV Cam	4:2:0	5:1	DV Intra-image	720 × 576	25
	4:1:1			720 × 480	
DVC Pro 25	4:2:0	5:1	DV Intra-image	720 × 576	25
	4:1:1			720 × 480	
DVC Pro 50	4:2:2	3.3:1	DV Intra-image	720 × 576	50
				720 × 480	
Betacam SX	4:2:2	10:1	MPEG-2, grupos de duas imagens	720 × 576 720 × 480	18

Tabela 8.5: Formatos de produção de vídeo.

#### Formatos de ficheiros de vídeo

Os ficheiros de vídeo que guardamos nos discos dos computadores ou integramos numa aplicação multimédia, ou que distribuímos por streaming sobre a Internet são frequentemente encapsulados num contentor que determina o tipo de fluxos susceptíveis de ser inseridos no ficheiro: fluxo de vídeo e de áudio comprimidos, mas também, subtítulos, menus e procedimentos de interacção.

As informações existentes nos cabeçalhos dos ficheiros indicam quais os codecs que permitem restituir os dados áudio e vídeo sob forma de sequências lisíveis. Também aí se encontram informações que dizem respeito à resolução das imagens, à frequência de representação, ou aos copyrights que lhe podem estar associados. É este contentor que dá a sua extensão específica ao ficheiro de vídeo guardado.

Os formatos de ficheiro de vídeo mais frequentemente encontrados são:

- AVI (Audio-Video Interleave): standard microsoft de vídeo digital para Windows. Um ficheiro .avi poderá conter um fluxo de vídeo associado a vários fluxos de áudio e aceita vários tipos de compressão: Cinepack, MPEG-4, MP3, AC3, etc.). Num ficheiro AVI tipo-1, o conjunto do fluxo vídeo/áudio é guardado sem modificações sob forma de fluxo AVI único. Num ficheiro AVI tipo-2, o fluxo é partido em dados áudio e vídeo distintos, num mesmo ficheiro AVI. O tipo-1 tem a vantagem de guardar os dados no seu formato de origem, enquanto o tipo-2 é compatível com os softwares de vídeo incapazes de tratar os ficheiros de tipo-1. Um ficheiro .avi pode também conter dados de vídeo não comprimido, no formato RAW, o que garante um armazenamento volumoso, mas sem perdas de qualidade.
- Quicktime: standard de vídeo digital para Macintosh (extensões .mov, .qt, .qtx, .qtr, .qt3). O software de leitura QuickTime é capaz de ler mais de 200 formatos de áudio e de vídeo, dos quais MPEG-1, MPEG-4, Soreson Video, AVI, DV, Flash, MP3, etc.
- OGM (Ogg Vorbis Compressed Video file) da Xiph.Org Foundation: desenvolvido em open source. A extensão .ogg é para áudio e a extensão .ogm é para vídeo.
- MKV da Matroska: ficheiro de extensão .mkv. É capaz de associar fluxos de áudio e de vídeo a uma gestão de subtítulos e de menus, assim como a uma rápida procura nos ficheiros.

#### **Formatos de ficheiros de vídeo ligados a um codec**

Estes formatos parecem-se com os anteriores, mas possuem uma diferença. Usam para a compressão dos dados um único tipo de codec. Em contrapartida, a sua inferior portabilidade é compensada em geral por uma optimização da compressão.

Apresentaremos aqui apenas os formatos mais usados:

- RealMedia, formato criado pela RealNetworks para o streaming sobre Internet. Extensões .rm, .ram, .rpm. Os algoritmos usados são propriedade da RealNetworks.
- ASF (Advanced Streaming Format) e WMV (Windows Media file with audio/Video): formato criado pela Microsoft para streaming de vídeo sobre Internet. Estes formatos ligados exclusivamente à MPEG-4 podem integrar fluxos vídeo e áudio, visualizadores assim como acontecimentos de sincronização.
- Formatos ligados à norma MPEG:
  - .mpg: ficheiro comprimido segundo a norma MPEG-1;
  - .m2p: ficheiro comprimido segundo a norma MPEG-2;
  - .m2v: ficheiro contendo apenas vídeo comprimido segundo a norma MPEG-2;
  - .mp4: ficheiro de vídeo comprimido segundo a norma MPEG-4;
  - .vob: ficheiro de vídeo tipo DVD;
  - etc.

### 8.3.9 MPEG-7

MPEG-7 tornou-se num standard em 2001 com o nome formal: "Multimedia Content Description Interface". O objectivo principal do MPEG-7 é servir a necessidade de procura baseada no conteúdo de áudio-visuais em aplicações como por exemplo livrarias digitais. No entanto, também poderá ser aplicado a aplicações multimédia envolvendo a geração e utilização de dados multimédia.

Os dados em MPEG-7 podem ser: imagens fixas, gráficos, modelos 3D, áudio, discurso, vídeo e informação referente à composição (como combinar os diferentes dados). Os dados em MPEG-7 podem ser representados em formato de texto, binário ou ambos.

#### Descrição de conteúdos multimédia

MPEG-7 desenvolveu descritores (D), esquemas de descritores (DS) e linguagem de definição de descrições (DDL). De seguida apresentamos alguns termos importantes:

**Característica** característica do dado.

**Ferramenta de Descrição** conjunto de Ds e DSs que descrevem a informação estrutural e conceptual do conteúdo.

**D (Descriptor)** definição (sintáctica e semântica) das características.

**DS (Esquema de descrição)** especificação da estrutura e relação entre os seus componentes, que podem ser Ds e DSs.

**DDL (linguagem de definição de descrição)** regras sintácticas para exprimir e combinar DSs e Ds. MPEG-7 adopta o XML que estende para satisfazer requerimentos ligados a conteúdos áudio-visuais.

O objectivo do MPEG-7 é estandardizar os Ds, DSs e DDL para a descrição dos conteúdos. O mecanismo e processo para produzir ou usar esses descritores sai fora destes do domínio do MPEG-7.

#### Descritores e esquema de descrição (Ds e DSs)

As descrições de conteúdo poderão incluir:

- informação descrevendo os processos de criação e produção do conteúdo (director, título, pequenas características do filme,...);
- informação relacionada com a utilização do conteúdo (copyright, historial de utilização,...);
- informação das ferramentas de armazenamento do conteúdo (formato de armazenamento, codificação,...);
- informação estrutural dos componentes espaciais, temporais ou espaço-temporais do conteúdo ( cortes da cena, segmentação em regiões, seguimento do movimento duma região, ...);
- informação sobre características de baixo nível do conteúdo (cor, texturas, timbres do som, descrição de melodias,...);

- informação conceptual da realidade capturada pelo conteúdo (objectos e acontecimentos, interacção entre objectos,...);
- informação sobre como navegar no conteúdo de forma eficiente (sumários, variações, sub-bandas espaciais e de frequência,...);
- informação sobre a interacção do utilizador com o conteúdo (preferências do utilizador, historial de utilização,...).

### **8.3.10 MPEG-21**

O objectivo do MPEG-21 é definir uma ferramenta multimédia que permita um maior e mais transparente uso de recursos multimédia através de várias redes e dispositivos usados pelas diferentes comunidades.

MPEG-21 é baseado em dois pontos essenciais:

- definição de uma unidade de distribuição e transacção fundamental (Item Digital);
- conceito de Utilizadores que interagem com Itens Digitais.