

Hélio Pedrini

William Robson Schwartz



# Análise de Imagens Digitais

Princípios, Algoritmos  
e Aplicações



## COMPRESSÃO DE IMAGENS

As técnicas de compressão de imagens surgiram devido à necessidade de se reduzir o espaço requerido para o armazenamento e o tempo necessário para a transmissão de imagens. Tipicamente, imagens ocupam grandes quantidades de memória. Por exemplo, uma imagem colorida com dimensões de  $1024 \times 1024$  pixels, cada pixel representado por 24 bits, requer 3 Mbytes para seu armazenamento sem compressão. Dessa forma, o armazenamento de um elevado número de imagens digitais pode acarretar em alto custo computacional, tais como em aplicações que manipulam imagens médicas, imagens de satélites e bases de vídeo.

A transmissão de imagens digitais demanda requisitos elevados em termos de armazenamento e largura de banda dos canais de comunicação. Por exemplo, um vídeo com duração de 1 minuto formado por imagens de  $512 \times 512$  pixels, exibidas a uma taxa de 30 imagens por segundo, cada pixel representado por 24 bits, requer aproximadamente 1.4 Gbytes para seu armazenamento. O uso de técnicas de compressão pode proporcionar grande economia em aplicações que demandam alto desempenho em tempo real, tais como em videoconferência, televisão digital, telemedicina e comunicação militar via satélite.

As técnicas de compressão de imagens normalmente são classificadas em duas categorias, *compressão sem perda* e *com perda*. Na compressão sem perda, a imagem resultante após o processo de descompactação é exatamente igual à imagem original. Esse tipo de compressão é utilizado em aplicações em que os dados são de difícil aquisição ou contêm informação que não deve ser alterada pelo processo de compressão, como em diagnóstico médico por imagens.

Na compressão com perda, nem toda a informação é recuperada após a descompactação da imagem. Aplicações típicas incluem a videoconferência e televisão digital, em que a perda de certas informações pode ser tolerada pelo receptor.

As técnicas de compressão de imagens são baseadas na redução de redundâncias que existem na representação dos dados. Há diversas maneiras de descrever a redundância em imagens digitais. Uma forma comum utiliza a correlação espacial existente entre os dados da imagem.

Este capítulo descreve as principais técnicas utilizadas na compressão de imagens. Conceitos básicos da teoria da informação, importantes para a determinação de limites para a representação da informação, são examinados. Os principais tipos de redundância que podem ser explorados em imagens são apresentados e discutidos.

## 6.1 Redundância de Dados

A *redundância de dados* é um conceito fundamental em compressão de imagens. *Informação* é a porção dos dados que deve ser preservada tal que a interpretação de seu significado ou propósito possa ser realizada corretamente. A informação contida em um conjunto de dados normalmente é medida em bits para propósitos de armazenamento ou em bits por segundo (Hertz) para propósitos de transmissão em um canal de comunicação.

A redundância de dados é uma entidade matematicamente quantificável. Se  $n_1$  e  $n_2$  denotam o número de unidades de transporte de informação (bits, por exemplo) em dois conjuntos de dados que representam a mesma informação, a taxa de compressão pode ser expressa por

$$C_R = \frac{n_1}{n_2} \quad (6.1)$$

A redundância de dados relativa  $R_D$  do primeiro conjunto de dados (aquele caracterizado por  $n_1$ ) pode ser definida como

$$R_D = 1 - \frac{1}{C_R} \quad (6.2)$$

Uma taxa de compressão de 10 (ou 10:1), por exemplo, significa que o primeiro conjunto de dados possui 10 unidades de transporte de informação para cada unidade no segundo conjunto ou, de forma similar, 90% dos dados no primeiro conjunto são redundantes, uma vez que o segundo conjunto transporta o mesmo volume de informações com apenas 10% dos dados utilizados no primeiro.

As redundâncias de dados que podem ser identificadas e exploradas em compressão de imagens são, em geral, classificadas em três categorias: redundâncias de codificação, interpixel e psicovisual. Técnicas de compressão de imagens visam à redução ou à eliminação dessas redundâncias.

### 6.1.1 Redundância de Codificação

Seja  $f$  uma imagem com  $L$  níveis de cinza. Cada um dos níveis de cinza  $i$ , em que  $i = 0, 1, \dots, L-1$ , ocorre na imagem com probabilidade  $p_i$ , tal que

$$p_i = \frac{n_i}{n} \quad (6.3)$$

em que  $n_i$  é o número de pixels de intensidade  $i$  e  $n$  é o número total de pixels na imagem.

O número médio de bits necessários para codificar cada pixel em uma imagem é dado pelo somatório do produto do número de bits  $l(i)$  utilizados para representar cada nível de cinza  $i$  e a probabilidade com que o nível de cinza ocorre na imagem, ou seja

$$\bar{l} = \sum_{i=0}^{L-1} l(i) p_i \quad (6.4)$$

Assim, o número total de bits necessários para codificar uma imagem com dimensões  $n = M \times N$  pixels é dada por  $MN\bar{l}$ . Um código é denominado *ótimo* se seu comprimento mínimo é igual a  $\bar{l}$ .

**Exemplo 6.1** Seja uma imagem com sete níveis de cinza distintos, conforme a distribuição de probabilidade mostrada na tabela 6.1

Tabela 6.1: Distribuição dos níveis de cinza em uma imagem monocromática.

| Nível de cinza (i) | 0   | 1   | 2   | 3    | 4    | 5    | 6     |
|--------------------|-----|-----|-----|------|------|------|-------|
| Probabilidade P(i) | 1/2 | 1/4 | 1/8 | 1/16 | 1/32 | 1/64 | 1/128 |

Para representar cada nível de cinza, como há sete níveis distintos, seriam necessários 3 bits. Entretanto, caso fosse utilizada a representação da tabela 6.2, seriam necessários

$$\bar{L} = \sum_{i=0}^6 l(i) p_i = 1(1/2) + 2(1/4) + 3(1/8) + 4(1/16) + 5(1/32) + 6(1/64) + 7(1/128) \approx \\ \approx 1.93\text{bits}$$

Tabela 6.2: Codificação dos níveis de cinza.

| Nível de cinza (i) | 0 | 1  | 2   | 3    | 4     | 5      | 6       |
|--------------------|---|----|-----|------|-------|--------|---------|
| Código             | 0 | 10 | 110 | 1110 | 11110 | 111110 | 1111110 |

A partir da equação 6.2, tem-se que a taxa de compressão resultante  $C_R$  é igual a 3/1.93 ou 1.55. Assim, o nível de redundância, determinado pela equação 6.1, é dado por

$$R_D = 1 - \frac{1}{1.55} \approx 0.35$$

■

A maneira de codificar os níveis de cinza da imagem pode gerar grandes diferenças na quantidade de informação requerida para representar a imagem. Se os códigos utilizados para representar os níveis de cinza tiverem um número de símbolos (ou seja, bits) maior que o valor absolutamente necessário, a imagem apresentará *redundância de codificação*.

Para ilustrar o conceito de redundância de codificação, seja uma imagem de dimensões  $320 \times 320$  pixels com apenas quatro níveis de cinza, os quais podem ser representados por quatro símbolos diferentes, A, B, C e D. O nível de cinza A aparece em 3/4 da imagem, o nível B aparece em 1/8 e cada um dos níveis C e D aparece em 1/16 da imagem. Como há quatro níveis de cinza, torna-se natural utilizar apenas dois bits para representar cada pixel, ou seja, o número total de bits utilizados neste caso é igual a 204800 bits, como mostra a tabela 6.3.

Tabela 6.3: Códigos de comprimento fixo.

| Nível de cinza | Codificação | Número de pixels | Número de bits |
|----------------|-------------|------------------|----------------|
| A              | 00          | 76800            | 153600         |
| B              | 01          | 12800            | 25600          |
| C              | 10          | 6400             | 12800          |
| D              | 11          | 6400             | 12800          |
|                |             |                  | Total: 204800  |

Entretanto, como alguns níveis de cinza são mais freqüentes que outros na imagem apresentada neste exemplo, cada nível de cinza possui uma probabilidade diferente de ocorrer, tal que representar cada pixel com dois bits pode não ser a melhor escolha.

Uma outra codificação possível seria utilizar menos bits para os níveis de cinza mais freqüentes (conseqüentemente, mais bits seriam utilizados para os níveis de cinza menos freqüentes), de modo a tentar reduzir o número de bits utilizados para representar toda a imagem. A tabela 6.4 mostra um exemplo de possível codificação.

Tabela 6.4: Códigos de comprimento variável.

| Nível de cinza | Codificação | Número de pixels | Número de bits |
|----------------|-------------|------------------|----------------|
| A              | 0           | 76800            | 76800          |
| B              | 10          | 12800            | 25600          |
| C              | 110         | 6400             | 19200          |
| D              | 111         | 6400             | 19200          |
|                |             |                  | Total: 140800  |

Na representação por códigos de comprimento variável, mesmo utilizando um número maior de bits para alguns níveis de cinza, a mesma imagem ocupa somente 68.75% do seu tamanho inicial, ou seja, foi obtida uma compactação um pouco acima de 10:7. Outra forma de explorar a redundância de codificação seria procurar, em todo o código, agrupamentos de símbolos que se repetem e, então, atribuir a eles um único símbolo novo.

Em geral, a redundância de codificação está presente quando os códigos atribuídos a um conjunto de eventos (tal como os níveis de cinza da imagem) não foram escolhidos de forma a explorar as probabilidades dos eventos.

### 6.1.2 Redundância Interpixel

Enquanto a redundância de codificação explora a proporção desbalanceada de cada símbolo, a *redundância interpixel* explora a característica de que pixels vizinhos em uma imagem normalmente possuem alguma relação ou similaridade.

Tipicamente, a maioria dos pixels possui valores não muito diferentes de seus vizinhos. Então, como o valor de cada pixel é razoavelmente previsível a partir dos valores dos seus vizinhos, a informação que cada pixel carrega é relativamente pequena. Muito da contribuição visual de um único pixel para uma imagem é redundante, pois ela poderia ser prevista com base nos valores dos pixels adjacentes.

Para aproveitar essa característica, a representação da imagem por uma matriz bidimensional de pixels, utilizada para visualização e interpretação humana, já não é mais um modelo eficiente. Torna-se necessário transformar a matriz de pixels em um formato mais adequado, evitando que as mesmas informações sejam associadas a cada pixel e seus vizinhos.

Uma possível representação seria armazenar apenas a diferença entre pixels adjacentes. Devido à tendência de pixels vizinhos possuírem valores próximos, essa diferença entre eles geraria, em sua maioria, valores pequenos. Esse fato aliado a algum método de redução da redundância de codificação, tal como a utilização de códigos de comprimento variável, reduziria ainda mais o número de bits necessários para armazenar uma imagem.

Um outro exemplo de representação que explora a redundância interpixel é a *codificação por comprimento de corrida*. Essa representação explora o fato de que algumas imagens possuem regiões com muitos pixels vizinhos idênticos. Conforme pode ser observado na figura 6.1, um grande número de pixels consecutivos pode possuir o mesmo valor, preto (0) ou branco (1).



Figura 6.1: Imagem binária e suas linhas.

A codificação por comprimento de corrida percorre cada linha da imagem e, em vez de armazenar o valor preto (0) ou branco (1) para cada pixel, armazena apenas a intensidade ou cor e o número de pixels iguais para cada grupo de pixels idênticos. Assumindo uma imagem com 320 pixels de largura e 280 pixels de altura, as primeiras linhas, que possuem apenas pixels pretos, seriam representadas por (0, 320) em que 0 é a cor preta e 320 é o número de pixels com esta cor.

A codificação por comprimento de corrida para a linha 100, destacada na figura 6.1, corresponderia à seqüência dada por

$$(1, 29)(0, 9)(1, 55)(0, 4)(1, 8)(0, 3)(1, 79)(0, 5)(1, 9)(0, 5)(1, 7)(0, 10)(1, 97)$$

portanto, mais compacta que a codificação que armazenaria um símbolo para cada um dos 320 pixels desta linha, dada por

### 6.1.3 Redundância Psicovisual

Outra característica importante que pode ser explorada na compressão de imagens é a imprecisão do sistema visual humano em perceber certos detalhes em uma imagem. Uma pequena diferença de intensidade luminosa entre duas áreas distintas de uma imagem, por exemplo, pode não ser percebida pelo olho humano.

Essas imprecisões resultam no fato de que o sistema visual humano não responde com a mesma sensibilidade a todas as informações visuais. Algumas informações possuem menor importância relativa do que outras no processamento visual. Tais informações, denominadas *psicovisualmente redundantes*, podem ser eliminadas sem prejudicar significativamente a percepção da imagem.

Essa redundância difere das demais por permitir que dados presentes na imagem possam ser eliminados de forma irreversível. Consequentemente, a imagem resultante não é igual à imagem original. A figura 6.2 mostra um exemplo de compactação baseada na redundância psicovisual, em que os tons de cinza de uma imagem monocromática são quantizados em quatro níveis diferentes de profundidade,  $L = 256$ ,  $L = 128$ ,  $L = 64$  e  $L = 32$ .

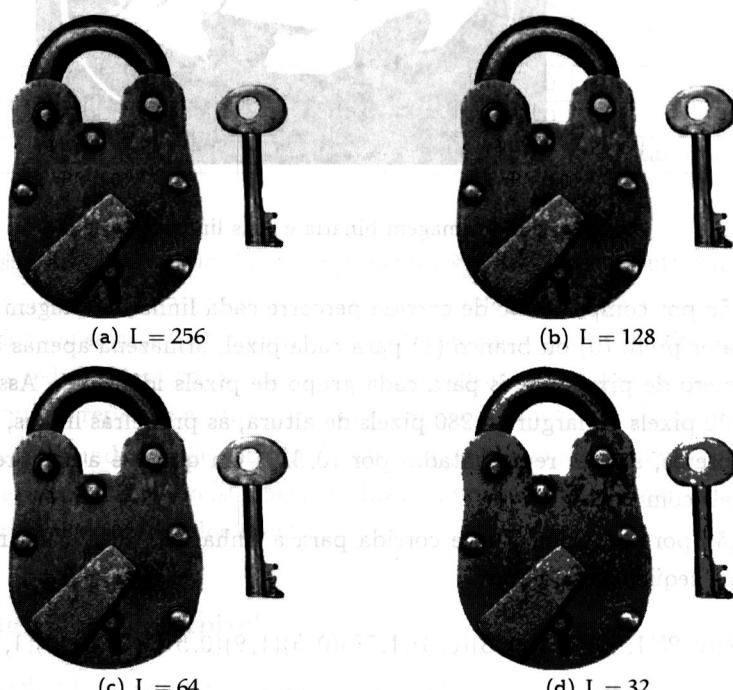


Figura 6.2: Redundância psicovisual.

## 6.2 Elementos de Teoria da Informação

A teoria da informação possibilita a determinação da quantidade mínima de dados necessária para representar uma imagem sem perda de informação.

A geração de informação pode ser modelada como um processo probabilístico, no qual um evento aleatório  $E$  que ocorre com probabilidade  $P(E)$  contém

$$I(E) = \log_b \frac{1}{P(E)} = -\log_b P(E) \quad (6.5)$$

unidades de informação. A quantidade  $I(E)$ , conhecida como *auto-informação* atribuída ao evento  $E$ , é inversamente proporcional à probabilidade de ocorrência de  $E$ . Se  $P(E) = 1$ , ou seja, o evento sempre ocorre, então  $I(E) = 0$  e, portanto, nenhuma informação é a ele

atribuída. Isso quer dizer que, pelo fato de não existir nenhuma incerteza associada ao evento, nenhuma informação precisaria ser transferida para comunicar que o evento ocorreu. Entretanto, se  $P(E) = 0.99$ , a comunicação de que E ocorreu transfere uma pequena quantidade de informação. Por outro lado, a comunicação de que E não ocorreu requer mais informação, pois esse resultado é menos provável.

A base do logaritmo na equação 6.5 determina a unidade utilizada para medir a informação. Se a base b for utilizada, a medida conterá b unidades. Caso a base 2 seja selecionada, a unidade resultante de informação é chamada de *bit*. Pode-se notar que, se  $P(E) = 1/2$ , então  $I(E) = -\log_2 1/2$ , isto é, 1 bit. Portanto, 1 bit corresponde à quantidade de informação transferida quando um dos dois eventos possíveis, igualmente prováveis, ocorre. Um exemplo simples de tal situação é a comunicação do resultado do lançamento de uma moeda.

Quando a informação  $I(E)$  é transferida entre uma fonte de informação e um usuário da informação, diz-se que a fonte está conectada ao usuário por um meio físico chamado de *canal de informação*. O canal pode ser uma linha telefônica, um cabo entre computadores ou um caminho de propagação de ondas eletromagnéticas. A figura 6.3 mostra um diagrama simples para um sistema de informações discreto.

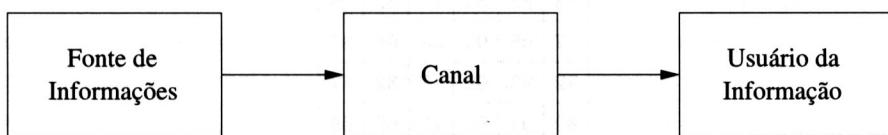


Figura 6.3: Sistema de informações.

Seja a fonte de informações apresentada na figura 6.3 que gera uma seqüência aleatória de símbolos a partir de um conjunto de símbolos possíveis. Em outras palavras, a saída da fonte é uma variável aleatória discreta. O conjunto de símbolos da fonte  $S = \{s_1, s_2, \dots, s_K\}$  é denominado *alfabeto-fonte* e os elementos do conjunto, denotados como  $s_i \in S$ , são chamados de *símbolos*. Uma fonte discreta possui um número finito de símbolos únicos. O número de elementos de um alfabeto S é chamado de cardinalidade.

A probabilidade de que a fonte produza o símbolo  $s_i$  é  $P(s_i)$ , tal que

$$\sum_{i=1}^K P(s_i) = 1 \quad (6.6)$$

O vetor  $\mathbf{v} = [P(s_1), P(s_2), \dots, P(s_K)]^T$ , de dimensões  $K \times 1$ , representa o conjunto de todas as probabilidades dos símbolos da fonte  $\{P(s_1), P(s_2), \dots, P(s_K)\}$ . O conjunto finito  $(S, \mathbf{v})$  descreve completamente a fonte de informações.

A probabilidade de que a fonte discreta emitirá o símbolo  $s_i$  é  $P(s_i)$ , assim, a autoinformação  $I(E)$  gerada pela produção de um único símbolo é, conforme a equação 6.5,  $I(s_i) = -\log P(s_i)$ . Se j símbolos-fontes forem gerados, a lei dos grandes números estipula que, para um valor suficientemente grande de j, o símbolo  $s_i$  será, em média, produzido  $jP(s_i)$  vezes. Assim, a informação média obtida a partir de j saídas é

$$-jP(s_1) \log P(s_1) - jP(s_2) \log P(s_2) - \dots - jP(s_K) \log P(s_K) \quad (6.7)$$

ou

$$-\sum_{i=1}^K P(s_i) \log P(s_i) \quad (6.8)$$

A informação média por saída da fonte, denotada  $H(v)$ , é definida como

$$H(v) = -\sum_{i=1}^K P(s_i) \log P(s_i) \quad (6.9)$$

e é denominada *incerteza* ou *entropia* da fonte. Ela define a quantidade média de informação obtida pela observação de uma única saída da fonte. Conforme a sua magnitude aumenta, mais incerteza e, portanto, mais informação estará associada com a fonte. Se os símbolos-fontes são igualmente prováveis, a entropia ou incerteza da equação 6.8 é maximizada e a fonte fornece a maior informação média possível por símbolo-fonte.

**Exemplo 6.2** Seja a imagem de  $6 \times 6$  pixels, com cinco níveis de cinza distintos, apresentada na figura 6.4. Portanto, a distribuição dos níveis de cinza na imagem é dada pelas probabilidades mostradas na tabela 6.5.

|    |    |    |    |    |    |
|----|----|----|----|----|----|
| 93 | 82 | 82 | 77 | 82 | 98 |
| 77 | 65 | 93 | 98 | 98 | 82 |
| 82 | 93 | 98 | 77 | 82 | 77 |
| 82 | 93 | 98 | 77 | 65 | 65 |
| 93 | 65 | 65 | 98 | 77 | 82 |
| 77 | 82 | 98 | 65 | 77 | 77 |

Figura 6.4: Imagem com cinco níveis de cinza distintos.

Tabela 6.5: Probabilidades de ocorrência dos níveis de cinza na imagem.

| Nível de cinza (i) | 0    | 1    | 2    | 3    | 4    |
|--------------------|------|------|------|------|------|
| $n_i$              | 6    | 9    | 9    | 5    | 7    |
| $p(i)$             | 0.17 | 0.25 | 0.25 | 0.14 | 0.19 |

Assim, a entropia da imagem f é dada por

$$H = -\sum_{i=0}^4 p_i \log_2 p_i = -(0.17 \log_2 0.17 + 0.25 \log_2 0.25 + 0.25 \log_2 0.25 + 0.14 \log_2 0.14 + 0.19 \log_2 0.19) \approx 2.29$$

Os conceitos anteriores são baseados no modelo mostrado na figura 6.3, que contém uma fonte de informações, canal e usuário. Um sistema de comunicação será agora adicionado ao modelo, tal que novos conceitos poderão ser apresentados sobre codificação ou representação da informação. O sistema de comunicação, inserido entre a fonte e o usuário, consiste em um codificador e um decodificador, conforme mostrado na figura 6.5.

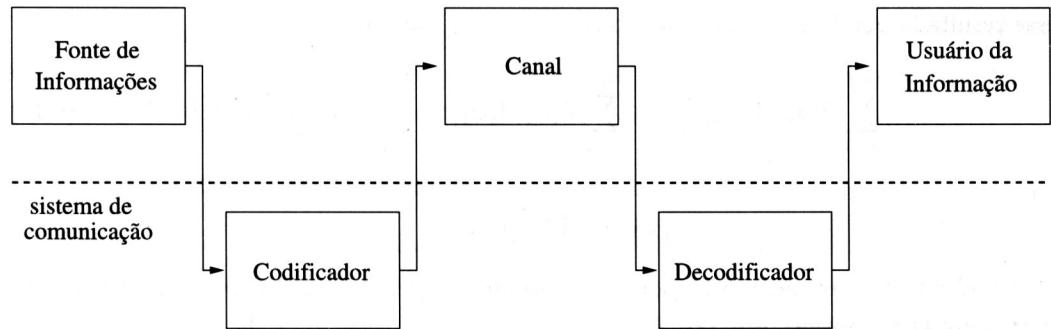


Figura 6.5: Sistema de comunicação.

Quando tanto o canal de informação quanto o sistema de comunicação são livres de erro, a principal função do sistema de comunicação é representar a fonte tão compactamente quanto possível. Sob essas circunstâncias, o *teorema da codificação sem ruído*, também conhecido como *primeiro teorema de Shannon* (Shannon 1948), define o comprimento mínimo médio da palavra de código por símbolo-fonte que pode ser alcançado.

Uma fonte de informações com conjunto finito  $(S, v)$  e símbolos-fontes estatisticamente independentes é chamado de fonte *sem memória*. Caso a saída seja considerada uma tupla de símbolos do alfabeto-fonte, a saída-fonte é uma variável bloco aleatória. Ela toma um dos valores  $K^n$  possíveis, denotados por  $s_i$ , do conjunto de todas as seqüências de  $n$  elementos possíveis. Cada  $\alpha_i$  é composto de  $n$  símbolos de  $S$ , que formam o conjunto de símbolos  $S' = \{\alpha_1, \alpha_2, \dots, \alpha_{K^n}\}$ .

A probabilidade de um dado  $\alpha_i$  é  $P(\alpha_i)$ , sendo relacionada às probabilidades de símbolos únicos  $P(\alpha_i)$  por

$$P(\alpha_i) = P(s_{i^1}) P(s_{i^2}) \dots P(s_{i^n}) \quad (6.10)$$

em que o subscrito adicional é utilizado para indexar os  $n$  símbolos de  $S$  que compõem um  $\alpha_i$ . Como anteriormente, o vetor  $v'$  denota o conjunto de todas as probabilidades  $\{P(\alpha_1), P(\alpha_2), \dots, P(\alpha_{k^n})\}$ . A entropia da fonte é dada por

$$H(v') = - \sum_{i=1}^{k^n} P(\alpha_i) \log P(\alpha_i) \quad (6.11)$$

Substituindo a equação 6.10 por  $P(\alpha_i)$  e simplificando, tem-se que

$$H(v') = nH(v) \quad (6.12)$$

Assim, a entropia da fonte de informação sem memória é  $n$  vezes a entropia da correspondente fonte de símbolo único. Como a auto-informação da saída-fonte  $\alpha_i$  é  $\log[1/P(\alpha_i)]$ , é razoável codificar  $s_i$  com uma palavra código de comprimento inteiro  $l(\alpha_i)$ , tal que

$$\log \frac{1}{P(\alpha_i)} \leq l(\alpha_i) < \log \frac{1}{P(\alpha_i)} + 1 \quad (6.13)$$

Intuitivamente, sugere-se que a saída-fonte  $\alpha_i$  seja representada por uma palavra código cujo comprimento seja o menor inteiro maior que a auto-informação de  $\alpha_i$ . Multiplicando-se

esse resultado por  $P(\alpha_i)$  e adicionando-se todos os  $i$ , resulta

$$\sum_{i=1}^{K^n} P(\alpha_i) \log \frac{1}{P(\alpha_i)} \leq \sum_{i=1}^{K^n} P(\alpha_i) l(\alpha_i) < \sum_{i=1}^{K^n} P(\alpha_i) \frac{1}{P(\alpha_i)} + 1 \quad (6.14)$$

ou

$$H(v') \leq \bar{L}' < H(v') + 1 \quad (6.15)$$

Dividindo-se a equação 6.15 por  $n$  e notando-se que, de acordo com a equação 6.14,  $H(v')/n$  é  $H(v)$ , então

$$H(v') \leq \frac{\bar{L}'}{n} < H(v) + \frac{1}{n} \quad (6.16)$$

A equação 6.16 enuncia o primeiro teorema de Shannon para uma fonte sem memória. Ela revela que é possível fazer  $\bar{L}'/n$  arbitrariamente próxima a  $H(v)$  por meio da codificação de extensões infinitamente longas da fonte. Embora derivada sob a hipótese de símbolos estatisticamente independentes, o resultado pode ser facilmente estendido para fontes mais gerais, como a fonte de Markov de  $m$ -ésima ordem, em que a ocorrência do símbolo fonte  $\alpha_i$  pode depender de um número finito  $m$  de símbolos anteriores. Como  $H(v)$  é um limite inferior de  $\bar{L}'/n$ , a eficiência  $\eta$  da codificação pode ser definida como

$$\eta = \frac{H(v)}{\bar{L}} \quad (6.17)$$

## 6.3 Métodos de Compressão de Imagens

Em muitas aplicações, a imagem após sua decodificação deve permanecer exatamente igual à imagem original, ou seja, não deve ocorrer perda de dados durante o processo de compressão. Métodos de *compressão sem perdas* permitem a codificação e decodificação de imagens livres de erros. Em aplicações médicas, por exemplo, a perda de informação pode comprometer a precisão de um diagnóstico.

Por outro lado, os métodos de *compressão com perdas* procuram elevar a taxa de compactação por meio da exclusão de parte da informação contida na imagem. Conforme discutido na seção 6.1.3, as codificações que exploram as redundâncias psicovisuais eliminam dados presentes nas imagens sem comprometer significativamente a qualidade das imagens.

### 6.3.1 Compressão sem Perdas

Há várias aplicações que requerem elevada quantidade de espaço de armazenamento em sua forma natural, por exemplo, o processamento e a análise de um número elevado de imagens de satélites, imagens médicas ou documentos digitalizados. Em muitas situações, uma pequena variação na intensidade de pixels, mesmo que imperceptíveis ao sistema visual humano, pode resultar em uma diferença significativa nos resultados da análise das imagens. Nesses casos, compressões são necessárias para reduzir o espaço de armazenamento, entretanto, perdas ou erros não podem ser admitidos durante o processo de codificação e decodificação das imagens.

Os métodos de compressão sem perdas exploram principalmente a redundância de codificação e a redundância interpíxel. As próximas seções apresentam algumas abordagens para compressão de imagens sem perdas.

### 6.3.1.1 Codificação de Huffman

A *codificação de Huffman* (Huffman 1952) explora apenas a redundância de codificação para compactar imagens. Conforme descrito na seção 6.1.1, cada símbolo (que representa, por exemplo, um nível de cinza) pode ser substituído por um código de comprimento variável, de forma que os símbolos que ocorrem mais freqüentemente na imagem recebem códigos menores que os códigos dos símbolos menos freqüentes. Para que cada símbolo seja identificado corretamente no processo de decodificação, nenhum código pode ser prefixo de outro código de comprimento maior. Códigos que satisfazem essa propriedade são conhecidos como *códigos livres de prefixo*.

O primeiro passo para a geração dos códigos de comprimento variável utiliza um processo chamado de *redução de fontes*, que ordena decrescentemente as probabilidades com que cada símbolo ocorre na imagem e agrupa os dois símbolos com menor probabilidade em um novo símbolo. Essa operação é repetida até que restem apenas dois símbolos. Esse processo é conhecido como *codificação de Huffman*.

A tabela 6.6 ilustra a codificação de Huffman para um conjunto de seis símbolos,  $S_1$  a  $S_6$ . Na redução de fonte, as duas menores probabilidades, 0.06 e 0.04, são combinadas para formar um símbolo de probabilidade 0.1. Esse símbolo e sua probabilidade são colocados na primeira coluna da redução de fontes, tal que as probabilidades da fonte reduzida são também ordenadas decrescentemente. O mesmo processo é aplicado às demais colunas.

Tabela 6.6: Etapa de redução de fonte na codificação de Huffman.

| Fontes Originais |               | Redução de Fontes |      |      |      |
|------------------|---------------|-------------------|------|------|------|
| Símbolo          | Probabilidade | 1                 | 2    | 3    | 4    |
| $S_2$            | 0.40          | 0.40              | 0.40 | 0.40 | 0.60 |
| $S_6$            | 0.30          | 0.30              | 0.30 | 0.30 | 0.40 |
| $S_1$            | 0.10          | 0.10              | 0.20 | 0.30 |      |
| $S_4$            | 0.10          | 0.10              | 0.10 |      |      |
| $S_3$            | 0.06          | 0.10              |      |      |      |
| $S_5$            | 0.04          |                   |      |      |      |

O segundo passo da codificação de Huffman atribui um código a cada fonte reduzida, iniciando na ordem inversa a que foram obtidas até a fonte original. Os símbolos 0 e 1 são atribuídos aos símbolos à direita na tabela 6.7. Adiciona-se um bit a cada símbolo previamente agrupado. Essa operação é repetida para cada fonte reduzida até que a fonte original seja atingida. O código de Huffman final é mostrado na terceira coluna da tabela 6.7.

Para esse exemplo, o comprimento médio do código é

$$\bar{L} = 0.40 * 1 + 0.30 * 2 + 0.10 * 3 + 0.10 * 4 + 0.06 * 5 + 0.04 * 5 = 2.2 \text{ bits/símbolo}$$

que é um valor próximo da entropia (2.14 bits/símbolo). De acordo com a equação 6.17, a eficiência do código de Huffman resultante é  $2.14/2.2 = 0.973$ .

Tabela 6.7: Atribuição de códigos de Huffman.

| Fontes Originais |               | Redução de Fontes |      |      |      |     |      |    |      |   |
|------------------|---------------|-------------------|------|------|------|-----|------|----|------|---|
| Símbolo          | Probabilidade | Código            | 1    |      | 2    |     | 3    |    | 4    |   |
| $S_2$            | 0.40          | 1                 | 0.40 | 1    | 0.40 | 1   | 0.40 | 1  | 0.60 | 0 |
| $S_6$            | 0.30          | 00                | 0.30 | 00   | 0.30 | 00  | 0.30 | 00 | 0.40 | 1 |
| $S_1$            | 0.10          | 011               | 0.10 | 011  | 0.20 | 010 | 0.30 | 01 |      |   |
| $S_4$            | 0.10          | 0100              | 0.10 | 0100 | 0.10 | 011 |      |    |      |   |
| $S_3$            | 0.06          | 01010             | 0.10 | 0101 |      |     |      |    |      |   |
| $S_5$            | 0.04          | 01011             |      |      |      |     |      |    |      |   |

A codificação de Huffman produz um código único a cada símbolo, o que permite a decodificação sem perda de informação. Para a seqüência 010100111100 codificada conforme o exemplo mostrado na tabela 6.7, a primeira palavra válida é 01010, que corresponde ao símbolo  $S_3$ . O próximo código válido é 011, que corresponde ao símbolo  $S_1$ . De maneira análoga para as próximas palavras, obtém-se a seqüência decodificada  $S_3S_1S_2S_2S_6$ . A técnica de Huffman é descrita no algoritmo 6.1.

### Codificação de Huffman

- 1 Os símbolos da fonte são listados em ordem decrescente de probabilidade.
- 2 Os dois símbolos com menor probabilidade recebem o valor 0 e 1. Esses dois símbolos são combinados em um novo símbolo, cuja probabilidade é igual à soma das duas probabilidades originais. A probabilidade do novo símbolo é ordenada de acordo com seu valor.
- 3 O procedimento é repetido até que os dois últimos símbolos recebam o valor 0 e 1.
- 4 O código para cada símbolo original da fonte é formado pelo percurso inverso na seqüência de valores 0 e 1.

Algoritmo 6.1: Codificação de Huffman.

Quando o número de símbolos que necessitam ser codificados é muito elevado, a geração dos códigos de Huffman torna-se um tarefa complexa do ponto de vista computacional. Além disso, códigos prohibitivamente longos poderão ser atribuídos aos símbolos menos freqüentes.

Uma variação da codificação de Huffman consiste em codificar apenas os  $m$  símbolos mais freqüentes, dentre os  $n$  símbolos da fonte. Para os demais símbolos, utiliza-se um código livre de prefixo, seguido de um código de comprimento fixo adequado. Essa variação é conhecida como *codificação de Huffman modificada*.

A tabela 6.8 ilustra a codificação de Huffman modificada para  $n = 21$  e  $m = 12$ . Nessa codificação, os símbolos  $s_1$  a  $s_{12}$  receberam os códigos de Huffman originais, enquanto os símbolos  $s_{13}$  a  $s_{21}$  utilizam um prefixo de 2 bits seguido de um código de comprimento fixo igual a 4 bits. O comprimento médio de um código modificado é de 4.24 bits/símbolo, que é um valor maior do que seria obtido com a codificação de Huffman, ou seja, 4.05 bits/símbolo. Entretanto, esse valor ainda é bastante próximo do limite teórico dado pela entropia da fonte, igual a 4.0 bits/símbolo.

Tabela 6.8: Codificação de Huffman modificada.

| Símbolo da Fonte | Probabilidade | Código de Huffman Modificado |
|------------------|---------------|------------------------------|
| s <sub>1</sub>   | 0.20          | 11                           |
| s <sub>2</sub>   | 0.10          | 011                          |
| s <sub>3</sub>   | 0.10          | 0000                         |
| s <sub>4</sub>   | 0.06          | 0101                         |
| s <sub>5</sub>   | 0.05          | 00010                        |
| s <sub>6</sub>   | 0.05          | 00011                        |
| s <sub>7</sub>   | 0.05          | 00100                        |
| s <sub>8</sub>   | 0.04          | 00101                        |
| s <sub>9</sub>   | 0.04          | 00110                        |
| s <sub>10</sub>  | 0.04          | 00111                        |
| s <sub>11</sub>  | 0.04          | 01000                        |
| s <sub>12</sub>  | 0.03          | 01001                        |
| s <sub>13</sub>  | 0.03          | 100000                       |
| s <sub>14</sub>  | 0.03          | 100001                       |
| s <sub>15</sub>  | 0.03          | 100010                       |
| s <sub>16</sub>  | 0.02          | 100011                       |
| s <sub>17</sub>  | 0.02          | 100100                       |
| s <sub>18</sub>  | 0.02          | 100101                       |
| s <sub>19</sub>  | 0.02          | 100110                       |
| s <sub>20</sub>  | 0.02          | 100111                       |
| s <sub>21</sub>  | 0.01          | 101000                       |

### 6.3.1.2 Codificação de Shannon-Fano

A codificação de Shannon-Fano (Shannon 1948, Fano 1949) é uma técnica bastante simples para geração de códigos livres de prefixo. Os símbolos da fonte são ordenados decrescentemente de acordo com os valores de probabilidade. A lista de símbolos é dividida em dois grupos, de forma que as somas das probabilidades dos dois grupos sejam aproximadamente iguais. Cada símbolo no primeiro grupo recebe o valor 0 como primeiro dígito do código, enquanto os símbolos do segundo grupo recebem o valor 1. Cada um desses grupos é então dividido de acordo com o mesmo critério e dígitos adicionais são acrescentados ao código. Esse processo é repetido até que cada subconjunto contenha apenas um símbolo. A técnica de Shannon-Fano é descrita no algoritmo 6.2.

Um exemplo da aplicação da técnica de Shannon-Fano é ilustrada na tabela 6.9. O resultado da codificação pode ser representado pela árvore binária mostrada na figura 6.6. O comprimento médio  $\bar{L}$  dos códigos resultantes satisfaz a equação 6.16, ou seja,  $H \leq \bar{L} < H+1$ .

Entretanto, a codificação de Shannon-Fano, diferentemente da codificação de Huffman, nem sempre produz códigos de prefixo ótimos. Um exemplo desse fato é ilustrado na tabela 6.10 para representar os símbolos dados pelas probabilidades {0.35, 0.17, 0.17, 0.16, 0.15}, em que os comprimentos médios dos códigos pela técnica de Huffman e de Shannon-Fano são, respectivamente, 2.30 e 2.31.

### Codificação de Shannon-Fano

- 1 Os símbolos da fonte são listados em ordem decrescente de probabilidade.
- 2 Os símbolos são divididos em dois grupos, de forma que as somas das probabilidades nos dois grupos sejam as mais próximas possíveis.
- 3 Um grupo recebe o valor 1 e o outro grupo recebe o valor 0.
- 4 O procedimento é repetido para os demais grupos até que reste apenas um símbolo em cada grupo.
- 5 O código para cada símbolo original da fonte é formado pela seqüência resultante de valores 0 e 1.

Algoritmo 6.2: Codificação de Shannon-Fano.

Tabela 6.9: Codificação de Shannon-Fano.

| Símbolo | Probabilidade | Código |
|---------|---------------|--------|
| $s_2$   | 0.30          | 0 0    |
| $s_6$   | 0.20          | 0 1    |
| $s_1$   | 0.20          | 1 0 0  |
| $s_4$   | 0.10          | 1 0 1  |
| $s_3$   | 0.10          | 1 1 0  |
| $s_5$   | 0.10          | 1 1 1  |

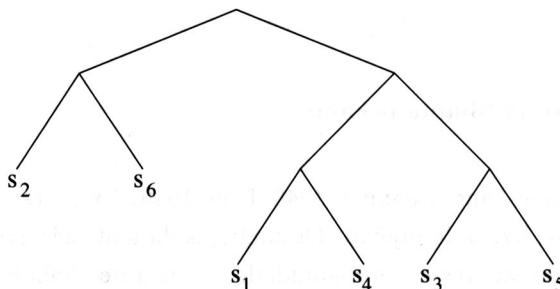


Figura 6.6: Árvore binária relativa à codificação Shannon-Fano para o exemplo mostrado na tabela 6.9.

#### 6.3.1.3 Codificação Aritmética

Na *codificação aritmética* (Rissanen 1976, Rissanen e Langdon 1979, Rubin 1979, Guazzo 1980), ao contrário da codificação de Huffman, não há correspondência individual entre os símbolos e os códigos. Conjuntos inteiros de símbolos são codificados em um único intervalo  $[0, 1]$ . O processo de codificação baseia-se em subdivisões sucessivas de intervalos numéricos, proporcionais às probabilidades de cada símbolo.

Seja, por exemplo, um determinado conjunto de dados possuindo apenas os símbolos A e B (tal como em uma imagem binária), em que a probabilidade de ocorrência do símbolo A é 0.6 e a do símbolo B é 0.4. Para a codificação da seqüência BAAB, basta dividir o conjunto inicial  $[0, 1]$  proporcionalmente à probabilidade de cada símbolo e escolher o

Tabela 6.10: Códigos obtidos pela técnica de Huffman e de Shannon-Fano para os símbolos com probabilidades  $\{0.35, 0.17, 0.17, 0.16, 0.15\}$ .

| Símbolo | Probabilidade | Código de Huffman | Código de Shannon-Fano |
|---------|---------------|-------------------|------------------------|
| $S_1$   | 0.35          | 00                | 1                      |
| $S_2$   | 0.17          | 01                | 011                    |
| $S_3$   | 0.17          | 10                | 010                    |
| $S_4$   | 0.16          | 110               | 001                    |
| $S_5$   | 0.15          | 111               | 000                    |

subintervalo correspondente ao primeiro símbolo para repetir o processo ao seguinte, e assim sucessivamente, como mostrado na figura 6.7.

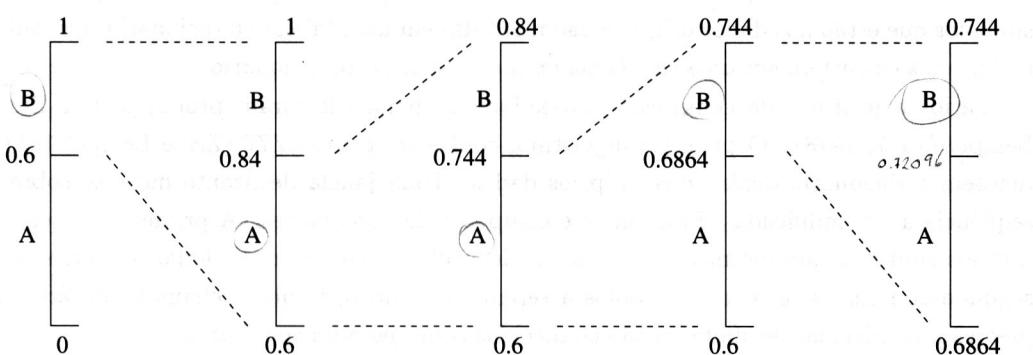


Figura 6.7: Codificação aritmética.  $BAAB \rightarrow [0.6864, 0.744]$

O resultado da decodificação de BAAB, nesse exemplo, é o intervalo  $[0.6864, 0.744]$ . Claramente, quanto maior o número de símbolos e quanto maior a seqüência a ser codificada, maior a precisão necessária para representar o intervalo resultante. Para decodificar o subintervalo resultante, basta verificar sucessivamente a qual intervalo o subintervalo resultante pertence, como mostrado na figura 6.8.

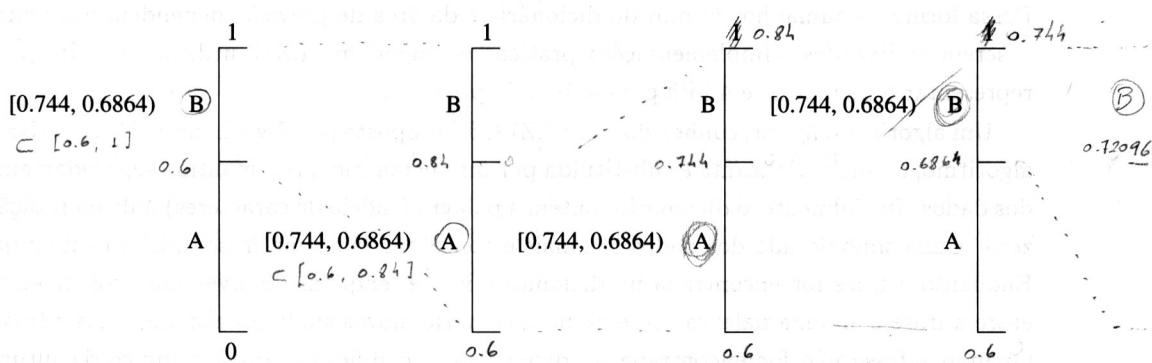


Figura 6.8: Decodificação aritmética.  $BAAB \rightarrow [0.6864, 0.744]$

Apesar de aparentemente muito eficiente, a codificação aritmética possui algumas características que degradam seu desempenho. Uma delas é o fato de ter que armazenar as probabilidades de cada símbolo para se obter na decodificação uma subdivisão exatamente igual à utilizada na codificação. Pode-se também considerar o número de bits utilizados para representar os intervalos como outro fator importante, já que, se utilizado um número fixo de bits para cada seqüência, este deverá ser suficientemente grande para que nenhuma seqüência perca precisão e, se utilizado um número variável de bits, há a necessidade de armazenar indicadores que separarão os diversos intervalos.

#### 6.3.1.4 Codificação Baseada em Dicionário

As técnicas de codificação apresentadas anteriormente requerem o cálculo das probabilidades dos símbolos da fonte. A *codificação baseada em dicionário*, por outro lado, não necessita de conhecimento sobre a freqüência com que os símbolos ocorrem nos dados originais.

Nas técnicas de compressão baseadas em dicionário, informações sobre a seqüência de símbolos que estão nos dados originais são mantidas em uma tabela ou dicionário. Os dados codificados consistem em uma seqüência de índices para este dicionário.

Vários algoritmos de compressão baseados em dicionário foram propostos por Ziv e Lempel (1977, 1978). O primeiro algoritmo, conhecido como LZ77 (Ziv e Lempel 1977), mantém o dicionário dentro dos próprios dados. Uma janela deslizante move-se sobre a seqüência a ser codificada. Essa janela é composta de duas partes. A primeira contém os últimos símbolos que foram codificados, sendo utilizada como um dicionário, enquanto a segunda contém os próximos símbolos a serem codificados, também chamada de área de previsão. Seqüências de símbolos são conhecidas como *palavras* ou *frases*.

O algoritmo consiste em procurar no dicionário a maior coincidência possível entre a frase composta pelos símbolos individuais da área de previsão e uma frase presente no dicionário. A codificação é realizada por uma tripla  $(i, n, s)$ , em que  $i$  é o índice do primeiro elemento da frase coincidente no dicionário,  $n$  é o número de elementos coincidentes e  $s$  é o próximo símbolo da área de previsão. Em seguida, a janela é deslocada de  $n + 1$  elementos e o processo se repete.

Quanto maior o tamanho do dicionário e da área de previsão, maior a probabilidade de se encontrar grandes coincidências. Entretanto, isso implica maior tempo de processamento para efetuar as buscas e maior número de bits dos componentes  $i$  e  $n$  na tripla de codificação. Dessa forma, os tamanhos ótimos do dicionário e da área de previsão dependem dos dados a serem codificados. Implementações práticas do algoritmo LZ77 utilizam 12 bits para representar o dicionário e 4 bits para a área de previsão.

Um algoritmo similar, conhecido como LZ78, foi proposto por Ziv e Lempel (1978). Nesse algoritmo, a janela deslizante é substituída por um dicionário que é mantido separadamente dos dados. Inicialmente, o dicionário contém a palavra (cadeia de caracteres) nula na posição zero. Cada símbolo lido dos dados originais é concatenado a uma frase, inicialmente nula. Enquanto a frase for encontrada no dicionário, isto é, enquanto houver uma coincidência entre a frase e alguma palavra presente no dicionário, novos símbolos são anexados à frase. Quando a frase não for encontrada no dicionário, o codificador envia o índice da última palavra encontrada no dicionário para a saída, seguido pelo último símbolo da frase. A palavra é então adicionada ao dicionário e anulada. O processo de leitura e anexação de

novos símbolos é repetido. Se a frase tiver um único símbolo e não for encontrada no dicionário, o codificador envia o símbolo para a saída precedido pelo índice 0 (índice da palavra nula). O número de bits do índice determina o número máximo de palavras que podem estar simultaneamente no dicionário.

Uma variação do algoritmo LZ78, conhecido como LZW, foi proposto por Welch (1984). O dicionário é inicialmente construído com todos os símbolos de comprimento 1 (símbolos individuais). O primeiro símbolo lido dos dados originais é anexado à frase, inicialmente nula, e então procura-se no dicionário a frase resultante. Evidentemente, esse primeiro símbolo está presente no dicionário. O próximo símbolo  $s$  dos dados é então concatenado à frase, que agora não se encontra no dicionário, sendo também nele inserida. Após a inserção da frase, o codificador envia para a saída o índice da última seqüência encontrada no dicionário. Reinicia-se, então, o processo com a frase igual a  $s$ . Novos símbolos lidos dos dados são concatenados à frase até que a seqüência resultante não seja mais encontrada no dicionário. Quando isso ocorrer, envia-se para a saída o índice da última seqüência encontrada e anexa-se a frase ao dicionário.

Na decodificação, o dicionário é inicialmente criado com os símbolos individuais dos dados. Então, cada seqüência recebida é traduzida por meio do dicionário em uma seqüência original. Exceto para o caso do primeiro símbolo, cada vez que uma seqüência é recebida, o dicionário é atualizado da seguinte forma: após a seqüência ter sido traduzida, seu primeiro símbolo é adicionado à frase anterior para acrescentar uma nova seqüência ao dicionário. Assim, o decodificador incrementalmente reconstrói o mesmo dicionário utilizado no codificador.

A codificação e a decodificação LZW são descritas nos algoritmos 6.3 e 6.4, respectivamente.

### Codificação LZW

```

1 frase f é inicialmente nula
2 para cada símbolo s da seqüência de entrada
3   se concatenação da frase com símbolo, fs, estiver no dicionário
4     então
5       símbolo é concatenado à frase, ou seja, f ← fs
6     senão
7       enviar índice do dicionário referente à frase f
8       inserir concatenação fs no dicionário
9       frase é formada pelo símbolo s, ou seja, f ← s

```

Algoritmo 6.3: Codificação LZW.

Um problema com os algoritmos LZ78 e LZW é o tamanho do dicionário. À medida que o número de entradas do dicionário aumenta, a realização de constantes buscas e inserções torna-se mais dispendiosa. Para minimizar esse problema, uma função de dispersão (do inglês, *hash function*) é utilizada para diretamente retornar um índice do dicionário a partir de uma dada chave de entrada.

**Exemplo 6.3** Seja uma fonte capaz de gerar os símbolos {A, B, C}. A codificação e a decodificação da seqüência de entrada ABABCBABAAAAAA pelo algoritmo LZW são ilustradas a seguir.

### Decodificação LZW

```

1  código_anterior ← primeiro código lido
2  enviar símbolo associado ao código_anterior
3  para cada código c de entrada
4      código_corrente ← c
5      se código_corrente estiver no dicionário
6          então
7              frase ← decodificação do código_corrente
8          senão
9              frase ← decodificação do código_anterior
10             s ← primeiro símbolo do código_anterior
11             frase ← concatenação do símbolo s à frase
12         enviar frase
13         s ← primeiro símbolo da frase
14         frase ← concatenação do símbolo s ao código_anterior
15         inserir frase no dicionário
16         código_anterior ← código_corrente

```

Algoritmo 6.4: Decodificação LZW.

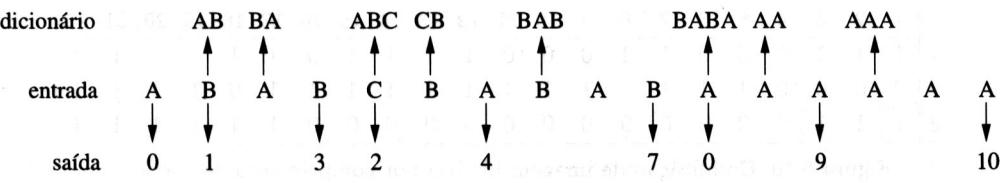
*O dicionário inicialmente conterá os símbolos da fonte {A, B, C}, cujos índices são dados pelos códigos 0, 1 e 2, respectivamente. Neste exemplo, os índices do dicionário estão representados em números decimais, em vez de códigos binários, para facilitar a compreensão da codificação. A seqüência de entrada é examinada, sendo que a palavra de maior comprimento para a qual existe uma frase correspondente no dicionário é extraída e o índice para essa frase é transmitido. A frase extraída é concatenada com mais um símbolo da seqüência de entrada, formando uma nova frase.*

*O primeiro símbolo lido da seqüência de entrada é A, o qual já se encontra no dicionário. O próximo símbolo é lido e concatenado à frase anterior. A frase resultante, AB, não se encontra no dicionário, então é nele inserida e o índice correspondente ao símbolo A é enviado para a saída. A frase é criada novamente, sendo composta apenas pelo último símbolo, ou seja B. O próximo símbolo lido é A que, concatenado na palavra anterior produz a palavra BA, que não se encontra no dicionário. O processo se repete até que toda a seqüência de entrada seja lida. Após a etapa de codificação, mostrada na figura 6.9(a), o dicionário conterá as entradas apresentadas na figura 6.9(b). A saída será formada pelos códigos (índices do dicionário) 0 1 3 2 4 7 0 9 10.*

*O processo de decodificação, mostrado na figura 6.9(c), também inicia com o dicionário formado com os três símbolos A, B e C, cujos códigos são, respectivamente, 0, 1 e 2. Para cada código recebido, o decodificador procurará o símbolo correspondente no dicionário e efetuará a decodificação. A etapa de decodificação recupera a informação original sem erros e reconstrói um dicionário idêntico ao da etapa de codificação.* ■

#### 6.3.1.5 Codificação por Comprimento de Corrida

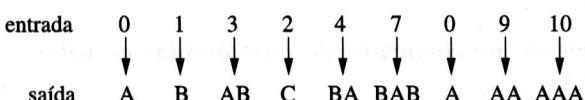
A codificação por comprimento de corrida explora a redundância interpíxel, de modo a armazenar, para cada seqüência de pixels iguais, apenas seu valor e o número de ocorrências.



(a)

| Índice  | 0 | 1 | 2 | 3  | 4  | 5   | 6  | 7   | 8    | 9  | 10  |
|---------|---|---|---|----|----|-----|----|-----|------|----|-----|
| Entrada | A | B | C | AB | BA | ABC | CB | BAB | BABA | AA | AAA |

(b)



(c)

Figura 6.9: Ilustração da codificação e da decodificação LZW. (a) codificação; (b) dicionário; (c) decodificação.

$A \rightarrow AB \rightarrow B \rightarrow BA \rightarrow A \rightarrow AB \rightarrow ABC \rightarrow C \rightarrow CB \rightarrow B \rightarrow BA \rightarrow BAB \rightarrow B \rightarrow BA \rightarrow BAB \rightarrow BABA \rightarrow \dots$

Portanto, a codificação por comprimento de corrida é mais apropriada em imagens cujos símbolos se repetem com grande freqüência, por exemplo, em imagens binárias.

Nessa codificação, a transformação da matriz bidimensional que representa a imagem em um vetor unidimensional é efetuada pela concatenação das linhas da imagem, de modo que o último pixel de uma linha preceda o primeiro pixel da linha seguinte. Assim, em uma imagem binária, as cores de cada corrida (ou agrupamento de pixels idênticos) são alternados, o que torna desnecessário o armazenamento da cor para cada corrida ou agrupamento.

Para o exemplo ilustrado anteriormente na figura 6.1, convencionando-se que todas as corridas em cada linha na imagem começem com a cor preta, o seguinte resultado é obtido para a linha 100

0, 29, 9, 55, 4, 8, 3, 79, 5, 9, 5, 7, 10, 97

em que o primeiro valor indica que a imagem possui 0 pixels pretos na primeira corrida. Isso significa, portanto, que a primeira corrida, de comprimento 29, é branca.

Na codificação por comprimento de corrida, os números que indicam os comprimentos das corridas de pixels podem ser utilizados para representar uma imagem binária. Se houver poucas corridas, essa codificação torna-se um método eficiente para reduzir o espaço de armazenamento requerido pela imagem.

Há duas abordagens comuns para codificação por comprimento de corrida. A primeira requer as posições iniciais e os comprimentos das corridas com valor 1 para cada linha. A segunda utiliza apenas os comprimentos das corridas, iniciando-se com o comprimento das corridas de valor 1.

**Exemplo 6.4 Codificação de imagem binária, mostrada na figura 6.10, por comprimento de corrida.**

A codificação efetuada pelas posições iniciais e os comprimentos das corridas com valor 1 é dada por

(1, 3), (7, 2), (12, 4), (17, 2), (20, 3)

|   | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0  | 0  | 1  | 1  | 1  | 1  | 0  | 1  | 1  | 0  | 1  | 1  | 1  |
| 2 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1  | 1  | 1  | 1  | 1  | 1  | 1  | 0  | 1  | 1  | 1  | 1  | 1  |
| 3 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0  | 0  | 0  | 0  | 0  | 0  | 0  | 1  | 1  | 1  | 1  | 1  | 1  |

Figura 6.10: Codificação de imagem binária por comprimento de corrida.

(5, 13), (19, 4)

(1, 3), (17, 6)

*A codificação baseada no comprimento das corridas de valor 1 é dada por*

3, 3, 2, 3, 4, 1, 2, 1, 3

0, 4, 13, 1, 4

3, 13, 6

■

Árvores binárias podem ser utilizadas para representar linhas de uma imagem na codificação por comprimento de corrida, quando o comprimento  $n$  é uma potência de 2, ou seja,  $n = 2^k$ . O nó raiz da árvore representa a linha inteira. Caso a linha apresente apenas um valor (0 ou 1), o nó raiz conterá um rótulo com esse valor e o processo termina; nesse caso, a árvore consiste em apenas o nó raiz. Caso contrário, dois descendentes são adicionados ao nó raiz, representando duas porções da linha. O processo é então repetido para cada um desses nós, ou seja, se cada uma das porções possuir um único valor, seu nó correspondente conterá um rótulo com esse valor. Caso contrário, dois novos descendentes são adicionados ao nó. Considerando que o nó raiz está no nível 0, os nós no nível  $h$  da árvore (caso existam) representam porções da linha de comprimento  $2^{k-h}$ . Se uma porção possuir valor constante, seu nó não possuirá descendentes (ou seja, é um nó folha), sendo rotulado com esse valor. Em um nível  $k$ , os nós (caso existam) correspondem a pixels únicos e são todos nós-folhas, rotulados com os valores de seus pixels. Um exemplo de uma linha de imagem e sua respectiva árvore binária é mostrada na figura 6.11.

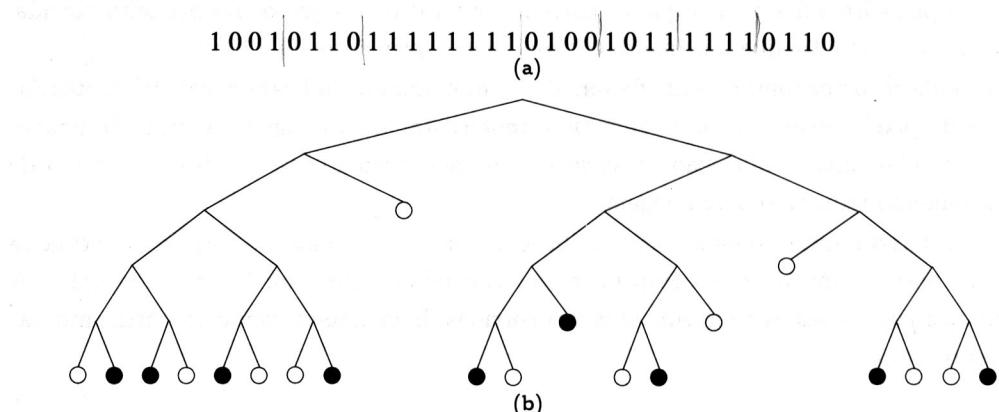


Figura 6.11: Representação de uma linha de imagem por meio de uma árvore binária. (a) linha de imagem binária com comprimento 32; (b) árvore binária correspondente, sendo que os círculos preenchidos e vazios são nós-folhas correspondendo a valores 1 e 0, respectivamente.

### 6.3.1.6 Codificação por Planos de Bits

A partir de uma imagem colorida ou mesmo monocromática, nem sempre se obtém uma boa taxa de compressão por meio da codificação por comprimento de corrida, pois a imagem normalmente apresenta um grande número de cores ou intensidades diferentes distribuídas em regiões pequenas na imagem.

A *codificação por planos de bits* é um conjunto de técnicas para decompor imagens multibandas em uma série de imagens binárias, chamadas de planos de bits, para que qualquer outra técnica de compressão para imagens binárias, como a codificação por comprimento de corrida, possa então ser utilizada.

Os níveis de cinza de uma imagem monocromática com  $m$  bits podem ser representados na forma de um polinômio de base 2

$$a_{m-1} 2^{m-1} + a_{m-2} 2^{m-2} + \dots + a_1 2^1 + a_0 2^0 \quad (6.18)$$

Com base nessa propriedade, um método de decomposição da imagem em uma coleção de imagens binárias consiste em separar os  $m$  coeficientes do polinômio em  $m$  planos de 1 bit. O plano de bits de ordem 0 é formado pelos coeficientes  $a_0$  de cada pixel, enquanto o plano de bits de ordem  $m - 1$  é formado pelos coeficientes  $a_{m-1}$ . Um exemplo dessa decomposição é ilustrado na figura 6.12, em que uma imagem monocromática com 8 bits por pixel é dividida em 8 imagens binárias, cada uma correspondendo à posição do seu respectivo bit da imagem original. Como observado na figura, quanto mais elevada a ordem dos planos de bits, maiores as áreas contínuas e uniformes.

A desvantagem inerente a essa abordagem é que pequenas mudanças nos níveis de cinza podem ter um impacto significativo na complexidade dos planos de bits. Como observado na figura 6.12(a), na imagem binária que representa o bit 7, toda transição entre o nível 127 (0111111) e o nível 128 (1000000) produz uma transição de 0 para 1 ou de 1 para 0 em todas as imagens binárias.

Métodos alternativos de decomposição procuram explorar mais adequadamente a redundância interpixel. Um deles é a codificação binária baseada no *código de Gray*, em que códigos sucessivos variam em apenas um bit de posição. As cinco palavras 000, 001, 011, 010 e 110, por exemplo, formam códigos de Gray de 3 bits. Assim, pequenas mudanças nos valores dos níveis de cinza afetam menos planos de bits, favorecendo a compressão das imagens. Uma vez decomposta a imagem original, métodos de compressão convencionais, como a codificação por comprimento de corrida, podem ser utilizados.

### 6.3.1.7 Codificação Preditiva sem Perdas

Alguns dos métodos descritos anteriormente, como a codificação de Huffman, exploram apenas a redundância de codificação. Uma abordagem importante para maximizar a compressão obtida nesse processo é a *codificação preditiva*, uma etapa que antecede a codificação de símbolos e tem como objetivo explorar a redundância interpixel.

Como descrito na seção 6.1.2, que trata da redundância interpixel, a simples transformação dos níveis de cinza em diferenças entre pixels adjacentes pode reduzir essa redundância pela eliminação, para cada pixel, das informações já presentes nos pixels anteriores.

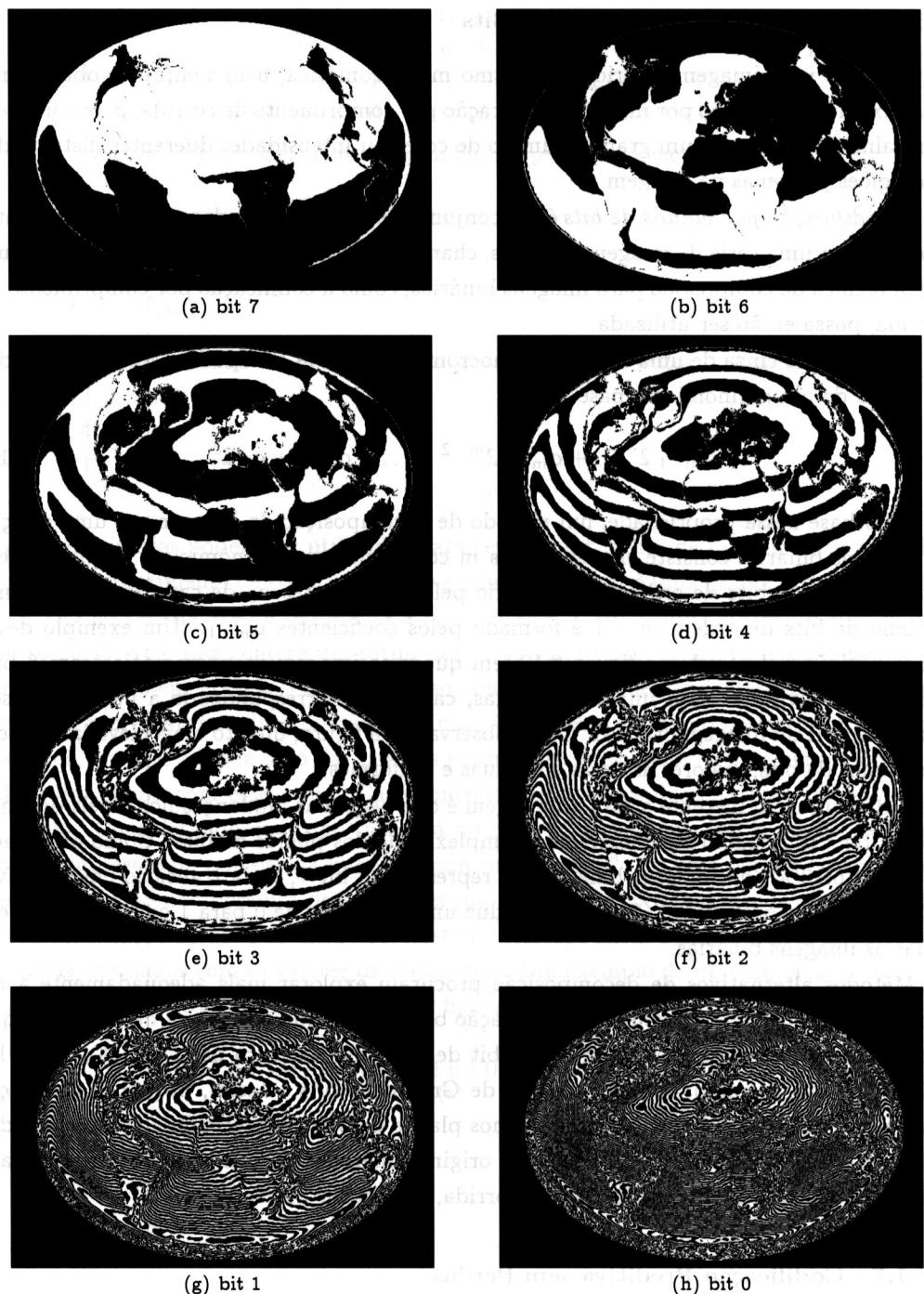


Figura 6.12: Planos de bits para bit 7, 6, 5, 4, 3, 2, 1 e 0.

A figura 6.13 apresenta um modelo de codificação preditiva sem perdas. O sistema consiste em um codificador e um decodificador, cada um deles contendo o mesmo *predictor*. À medida que cada pixel sucessivo em uma imagem, denotado por  $f_n$ , é introduzido no codificador, o predictor gera um valor antecipado daquele pixel com base em um dado número de entradas anteriores. A saída do predictor é então arredondada para o inteiro mais próximo,

denotado por  $\hat{f}_n$ , e usada para formar o erro de predição

$$e_n = f_n - \hat{f}_n \quad (6.19)$$

que é utilizado pelo codificador de símbolos para gerar o próximo elemento dos dados comprimidos. O decodificador, mostrado na figura 6.13(b), reconstrói  $e_n$  a partir da sequência de códigos de comprimento variável e realiza a operação inversa

$$f_n = e_n + \hat{f}_n \quad (6.20)$$

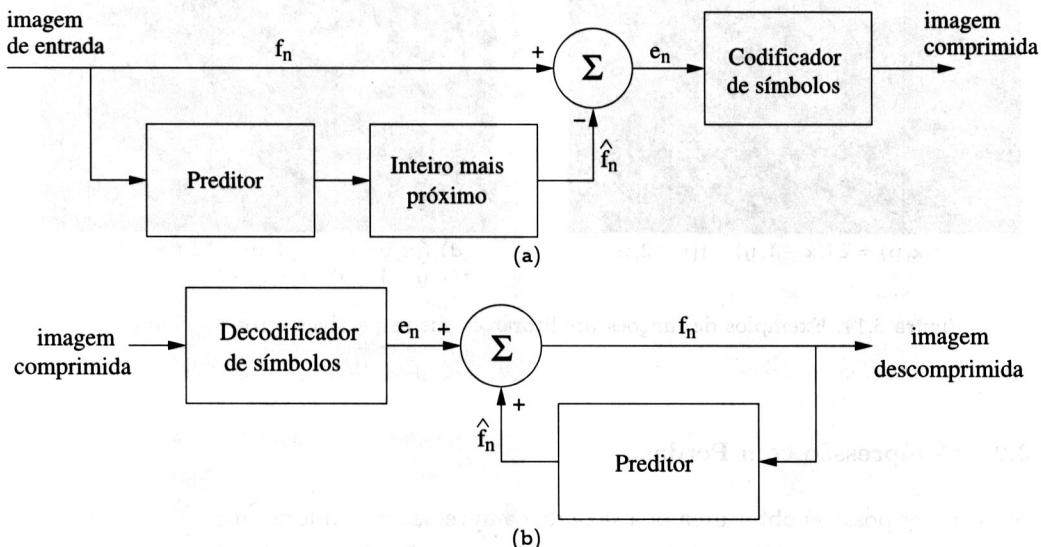


Figura 6.13: Codificação preditiva sem perdas. (a) codificador; (b) decodificador.

Desse modo, para cada pixel é gerada uma predição de que seu sucessor possui o mesmo valor. Ao armazenar apenas a diferença entre a predição e o valor real, quanto mais precisa for a predição, menor o valor a ser armazenado. Várias equações podem ser utilizadas na geração de  $\hat{f}_n$ . Na maioria dos casos, entretanto, a predição é formada por uma combinação linear de pixels anteriores. Alguns exemplos de funções preditoras são

$$\begin{aligned} \hat{f}(x, y) &= f(x - 1, y) \\ \hat{f}(x, y) &= 0.5 f(x - 1, y) + 0.5 f(x, y - 1) \\ \hat{f}(x, y) &= 2 f(x - 1, y) - f(x - 2, y) \\ \hat{f}(x, y) &= f(x - 1, y) - 0.5 f(x - 2, y) + f(x, y - 1) - 0.5 f(x, y - 2) \end{aligned} \quad (6.21)$$

A partir da imagem original da figura 6.12, aplicações das equações descritas em 6.21 são apresentadas na figura 6.14. O refinamento dessas equações pode levar a erros de predição menores e, consequentemente, a uma maior taxa de compressão.

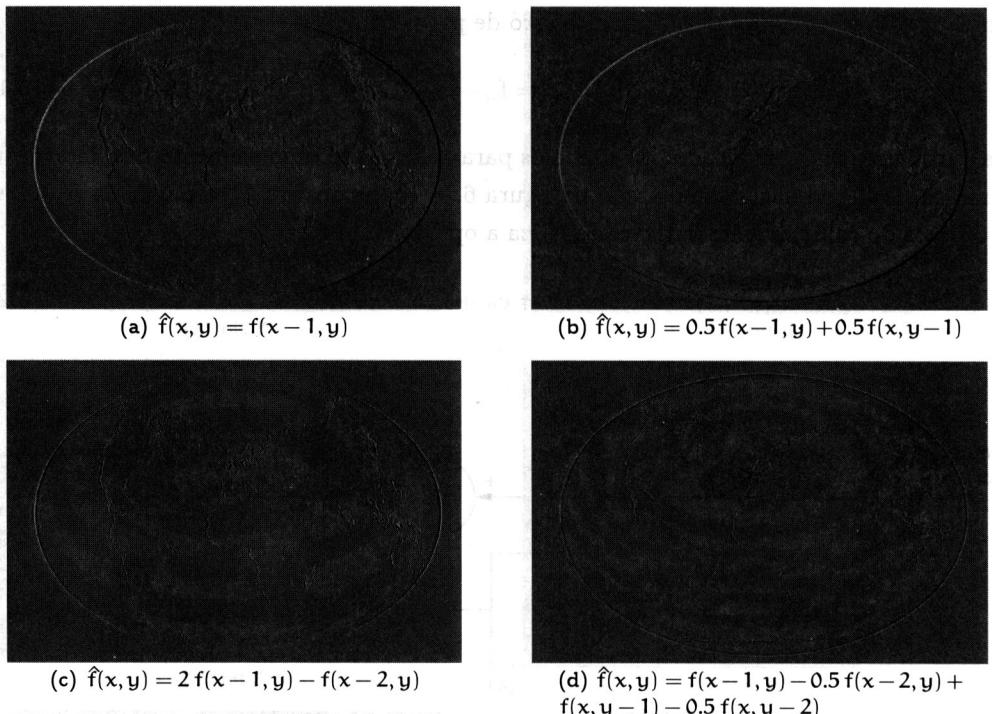


Figura 6.14: Exemplos de funções preditoras e seus respectivos erros de predição.

### 6.3.2 Compressão com Perdas

Apesar de ser possível obter uma boa taxa de compressão ao explorar apenas a redundância de codificação e a redundância interpíxel, há ainda a redundância psicovisual que, aliada às redundâncias utilizadas na compressão sem perdas, pode produzir imagens muito próximas às originais e ainda elevar a taxa de compressão em dezenas de vezes.

Em muitas aplicações, as distorções resultantes da eliminação de detalhes da imagem podem ser visualmente imperceptíveis ou toleradas. Nos métodos de compressão com perdas, não há um limite previamente estabelecido para a taxa de compressão, pois quanto mais detalhes são excluídos ou aproximados durante a codificação, maior será a compressão. O que há, portanto, é um compromisso entre a taxa de compressão e as perdas ou deformações na imagem resultante, como pode ser visto na figura 6.15, que apresenta uma imagem comprimida com diversas taxas de compressão. Uma ampliação de uma pequena área é apresentada à direita de cada imagem resultante.

#### 6.3.2.1 Codificação Preditiva com Perdas

Do mesmo modo que a codificação preditiva sem perdas prevê o valor do próximo pixel baseado nos pixels anteriores, a codificação com perdas prevê o próximo pixel, porém, antes de prosseguir, utiliza uma função quantizadora que exclui os dados poucos significativos e repassa essa informação para a função preditora, para que esta não acumule a perda nas próximas previsões.

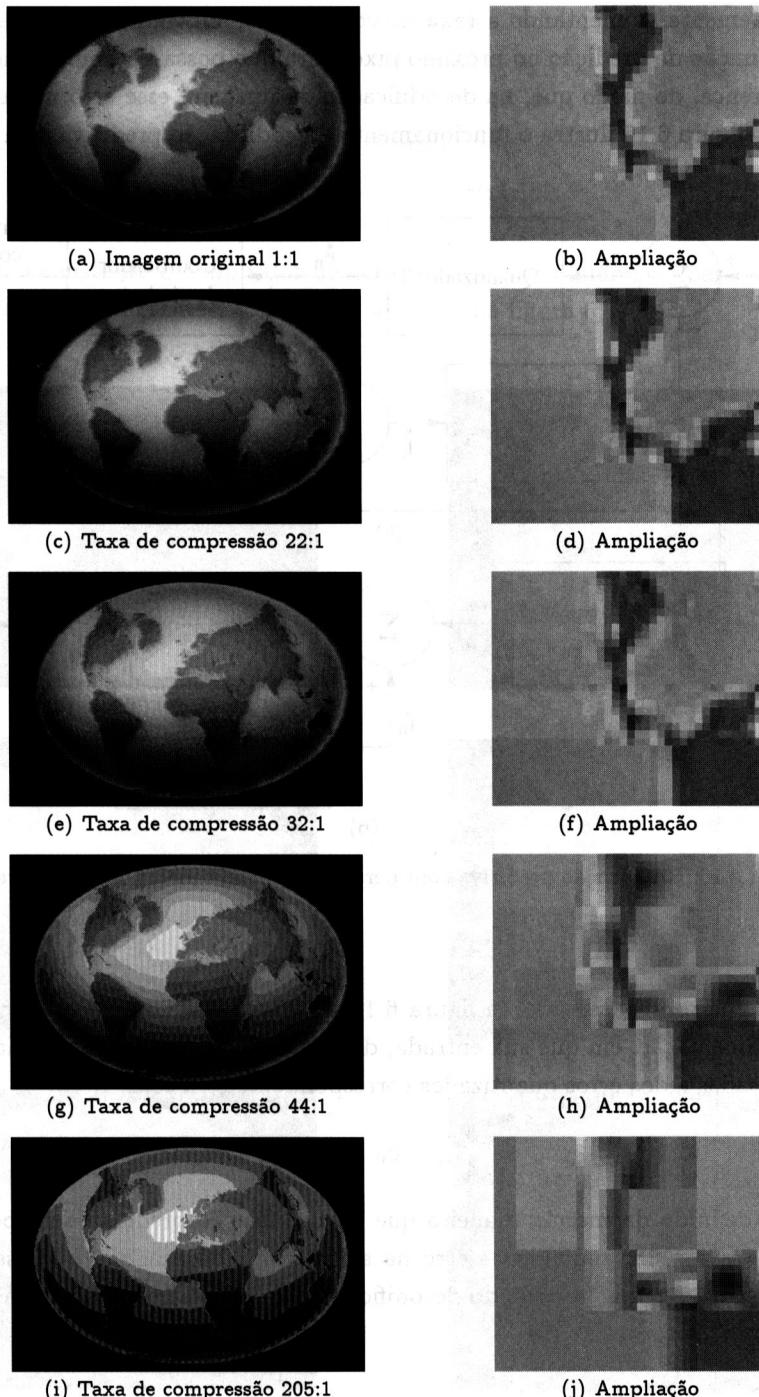


Figura 6.15: Compressão com perdas e deformações.

Um exemplo bastante simples dessa forma de codificação seria uma função preditora que retornasse apenas a diferença entre a predição e o valor real do pixel, como já visto anteriormente, para a função quantizadora. Esta, por sua vez, poderia considerar quaisquer diferenças pequenas como se fossem zero, diminuindo a quantidade de dados armazenados

e, consequentemente, aumentando a taxa de compressão. Obviamente, esse erro deve ser repassado à função de predição no próximo pixel para que possa ser somada novamente na próxima diferença, de modo que, na decodificação da imagem, esse erro não se acumule a cada pixel. A figura 6.16 ilustra o funcionamento da codificação preditiva com perdas.

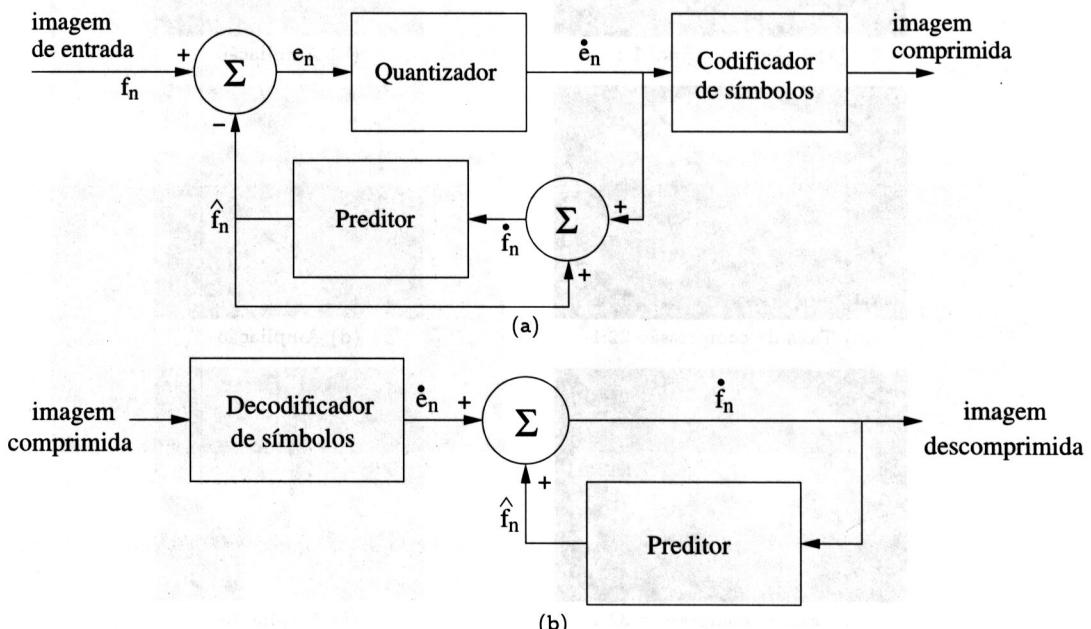


Figura 6.16: Codificação preditiva com perdas. (a) codificador; (b) decodificador.

Conforme pode ser observado na figura 6.16(a), o preditor com perdas do codificador possui retroalimentação, em que sua entrada, denotada  $\hat{f}_n$ , é gerada como uma função das previsões passadas e dos erros quantizados correspondentes, ou seja

$$\hat{f}_n = \dot{e}_n + \hat{f}_n \quad (6.22)$$

em que  $\hat{f}_n$  é definido da mesma maneira que na codificação preditiva sem perdas. Essa configuração previne um acúmulo de erro na saída do decodificador. Pode-se observar a partir da figura 6.16(b) que a saída do decodificador também é dada pela equação 6.22.

## Modulação Delta

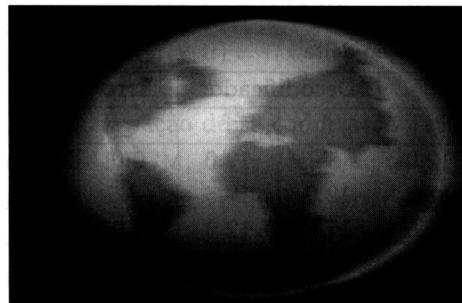
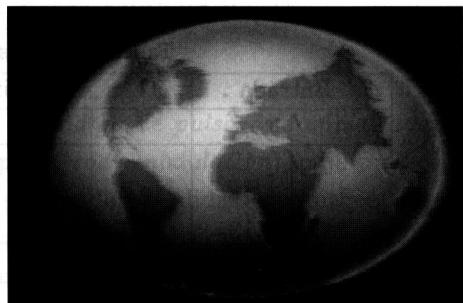
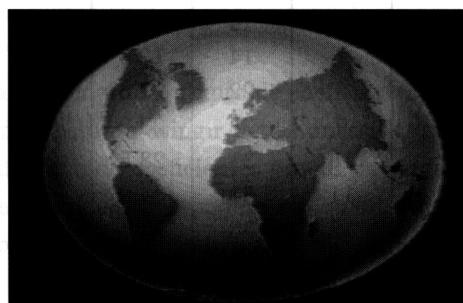
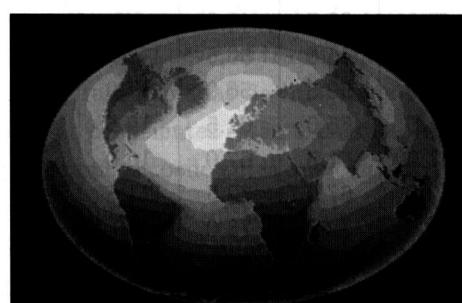
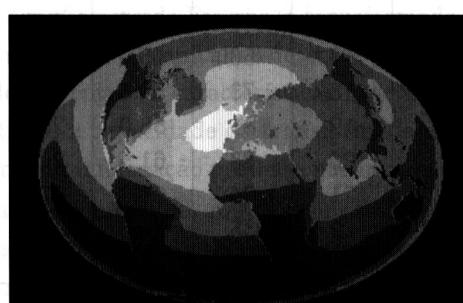
Um esquema simples de codificação preditiva com perdas é a *modulação delta* (DM, do inglês, *delta modulation*). Nessa codificação, o preditor supõe que o pixel atual seja igual ao anterior e, para cada pixel, retorne apenas a diferença entre este e o pixel anterior. A etapa quantizadora recebe essa diferença e retorna apenas dois valores possíveis:  $+\zeta$  para diferenças positivas e  $-\zeta$  para diferenças negativas. Essas considerações são expressas nas

equações 6.23 e 6.24.

$$\hat{f}_n = \dot{f}_{n-1} \quad (6.23)$$

$$\dot{e}_n = \begin{cases} +\zeta, & \text{para } e_n > 0 \\ -\zeta, & \text{caso contrário} \end{cases} \quad (6.24)$$

Em geral, a constante  $\zeta$  é um valor menor que 1. Valores diferentes para essa constante produzem resultados diferentes, como pode ser visto na figura 6.17.

(a)  $\zeta = 2$ (b)  $\zeta = 4$ (c)  $\zeta = 6$ (d)  $\zeta = 8$ (e)  $\zeta = 16$ (f)  $\zeta = 32$ Figura 6.17: Modulação delta para  $\zeta = 2, 4, 6, 8, 16$  e  $32$ .

Diferentes variações dessa técnica podem ser desenvolvidas, até mesmo o uso de mais de uma constante para que a precisão dos níveis de cinza seja melhorada, mas a principal vantagem dessa forma de codificação é que, com o quantizador retornando sempre  $+\zeta$  ou  $-\zeta$ , o resultado pode ser tratado e comprimido como uma imagem binária.

**Exemplo 6.5 Ilustração do processo de modulação delta para codificar e decodificar a seqüência de entrada {12, 14, 13, 16, 14, 12, 14, 14, 22, 28, 31, 31, 33, 33, 35, 39, 45, 58, 73, 81, 83, 84, 85, 85}, cuja função de previsão é  $\hat{f}_n = \hat{f}_{n-1} + \zeta$ , com  $\zeta = 5.5$ .**

O processo inicia-se com a transferência do primeiro pixel de entrada para o decodificador. Uma vez estabelecida a condição inicial  $\hat{f}_0 = f_0 = 12$ , tanto no codificador quanto no decodificador, as saídas subsequentes podem ser calculadas por meio das equações 6.19, 6.22, 6.23 e 6.24. Assim, quando  $n = 1$ ,  $\hat{f} = 12$ ,  $e_1 = 14 - 12 = 2$ ,  $\hat{e}_1 = +5.5$ ,  $\hat{f}_1 = 5.5 + 12 = 17.5$  e o erro de decodificação é  $14 - 17.5 = -3.5$  níveis de cinza. Os resultados completos do cálculo para a modulação delta são mostrados na tabela 6.11.

Tabela 6.11: Ilustração da modulação delta.

| Entrada |    | Codificador |      |           |      | Decodificador |      | Erro          |
|---------|----|-------------|------|-----------|------|---------------|------|---------------|
| n       | f  | $\hat{f}$   | e    | $\hat{e}$ | f    | $\hat{f}$     | f    | $f - \hat{f}$ |
| 0       | 12 | —           | —    | —         | 12.0 | —             | 12.0 | 0.0           |
| 1       | 14 | 12.0        | 2.0  | 5.5       | 17.5 | 12.0          | 17.5 | -3.5          |
| 2       | 13 | 17.5        | -4.5 | -5.5      | 12.0 | 17.5          | 12.0 | 1.0           |
| 3       | 16 | 12.0        | 4.0  | 5.5       | 17.5 | 12.0          | 17.5 | -1.5          |
| 4       | 14 | 17.5        | -3.5 | -5.5      | 12.0 | 17.5          | 12.0 | 2.0           |
| 5       | 12 | 12.0        | 0.0  | -5.5      | 6.5  | 12.0          | 6.5  | 5.5           |
| 6       | 14 | 6.5         | 7.5  | 5.5       | 12.0 | 6.5           | 12.0 | 2.0           |
| 7       | 14 | 12.0        | 2.0  | 5.5       | 17.5 | 12.0          | 17.5 | -3.5          |
| 8       | 22 | 17.5        | 4.5  | 5.5       | 23.0 | 17.5          | 23.0 | -1.0          |
| 9       | 28 | 23.0        | 5.0  | 5.5       | 28.5 | 23.0          | 28.5 | -0.5          |
| 10      | 31 | 28.5        | 2.5  | 5.5       | 34.0 | 28.5          | 34.0 | -3.0          |
| 11      | 31 | 34.0        | -3.0 | -5.5      | 28.5 | 34.0          | 28.5 | 2.5           |
| 12      | 33 | 28.5        | 4.5  | 5.5       | 34.0 | 28.5          | 34.0 | -1.0          |
| 13      | 33 | 34.0        | -1.0 | -5.5      | 28.5 | 34.0          | 28.5 | 4.5           |
| 14      | 35 | 28.5        | 6.5  | 5.5       | 34.0 | 28.5          | 34.0 | 1.0           |
| 15      | 39 | 34.0        | 5.0  | 5.5       | 39.5 | 34.0          | 39.5 | -0.5          |
| 16      | 45 | 39.5        | 5.5  | 5.5       | 45.0 | 39.5          | 45.0 | 0.0           |
| 17      | 58 | 45.0        | 13.0 | 5.5       | 50.5 | 45.0          | 50.5 | 7.5           |
| 18      | 73 | 50.5        | 22.5 | 5.5       | 56.0 | 50.5          | 56.0 | 17.0          |
| 19      | 81 | 56.0        | 25.0 | 5.5       | 61.5 | 56.0          | 61.5 | 19.5          |
| 20      | 83 | 61.5        | 21.5 | 5.5       | 67.0 | 61.5          | 67.0 | 16.0          |
| 21      | 84 | 67.0        | 17.0 | 5.5       | 72.5 | 67.0          | 72.5 | 11.5          |
| 22      | 85 | 72.5        | 12.5 | 5.5       | 78.0 | 72.5          | 78.0 | 7.0           |
| 23      | 85 | 78.0        | 7.0  | 5.5       | 83.5 | 78.0          | 83.5 | 1.5           |

### Modulação de Código de Pulso Diferencial

A técnica *modulação de código de pulso diferencial* (DPCM, do inglês, *differential pulse code modulation*) utiliza um preditor ótimo, cujo objetivo é minimizar o erro médio quadrático da predição do codificador.

Nessa codificação, assume-se que o erro de quantização seja desprezível, ou seja,  $\hat{e}_n \approx e_n$ , e que o valor previsto de um pixel dependa de uma combinação linear de  $m$  pixels anteriores. Essas restrições não são essenciais, entretanto, simplificam consideravelmente a análise e reduzem a complexidade computacional do preditor.

Alguns exemplos de funções preditoras são

$$\begin{aligned}\hat{f}(x, y) &= 0.97 f(x - 1, y) \\ \hat{f}(x, y) &= 0.5 f(x - 1, y) + 0.5 f(x, y - 1) \\ \hat{f}(x, y) &= 0.90 f(x - 1, y) - 0.81 f(x - 1, y - 1) + 0.90 f(x, y - 1) \\ \hat{f}(x, y) &= 0.75 f(x - 1, y) - 0.50 f(x - 1, y - 1) + 0.75 f(x, y - 1) \\ \hat{f}(x, y) &= f(x - 1, y) - f(x - 1, y - 1) + f(x, y - 1)\end{aligned}\quad (6.25)$$

Uma variação dessa codificação é a *modulação de código de pulso diferencial adaptativa* (ADPCM, do inglês, *adaptive differential pulse code modulation*), cuja predição adaptativa procura reduzir o erro de quantização, melhorando a qualidade da imagem decodificada. A quantização adaptativa é realizada de acordo com estatísticas locais e levando-se em conta, por exemplo, bordas presentes na imagem.

### 6.3.2.2 Codificação por Transformada

A codificação por transformada visa decompor a imagem em um conjunto de coeficientes, os quais podem ser posteriormente quantizados e codificados para atingir determinadas taxas de compressão. De maneira geral, um número significativo de coeficientes possui pequenas magnitudes, ou seja, representam pequenos detalhes da imagem e podem ser quantizados com maior taxa de erro ou mesmo descartados completamente com pouca distorção na imagem decodificada.

Diversas transformadas são comumente utilizadas em compressão de imagens, por exemplo, a transformada discreta do cosseno, a transformada discreta de Fourier, a transformada de Karhunen-Loëve, as transformadas wavelets, dentre outras. O capítulo 3 descreve as principais transformadas de imagens e suas propriedades. Dentre as características desejáveis para os coeficientes estão a baixa correlação estatística e a capacidade da transformada em gerar poucos coeficientes que concentrem grande parte da informação presente nas imagens.

A figura 6.18 ilustra as etapas fundamentais para a compressão por transformada. O codificador realiza quatro operações principais, a decomposição da imagem em um número de subimagens, o cálculo da transformada, a quantização e a codificação dos símbolos. Inicialmente, uma imagem com dimensões  $N \times N$  pixels é subdividida em subimagens de  $n \times n$  pixels. O cálculo da transformada visa reduzir a correlação entre os pixels de cada subimagem, de modo a compactar as informações em um número reduzido de coeficientes. O processo de quantização seleciona ou elimina os coeficientes que carregam menos informação da imagem. Esses coeficientes apresentam menor influência na qualidade da subimagem decodificada. A codificação dos símbolos normalmente é baseada no uso de códigos de comprimento variável para os coeficientes quantizados. Algumas ou todas as etapas da codificação por transformada podem ser adaptadas ao conteúdo local da imagem, ao que se denomina *codificação adaptativa por transformada*.

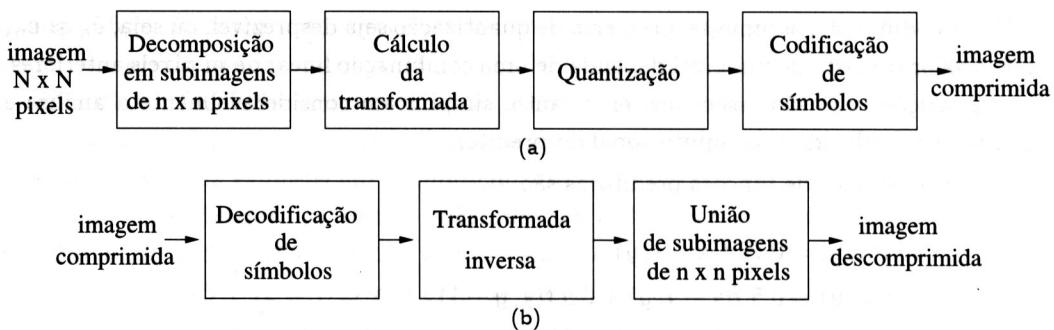


Figura 6.18: Compressão por transformada. (a) codificação; (b) decodificação.

Embora as transformadas pudessem ser realizadas considerando-se diretamente todos os pixels da imagem, essa abordagem geralmente não é adequada devido ao conteúdo não uniforme da imagem. Além disso, operações de transformação tornam-se menos custosas computacionalmente quando realizadas em porções menores da imagem. Assim, as imagens são decompostas em blocos de pixels. Há um compromisso entre o tamanho dos blocos e a complexidade computacional das transformadas, entretanto, tamanhos típicos de blocos são  $8 \times 8$  ou  $16 \times 16$  pixels.

Conforme descrito na seção 3.1.2, uma imagem  $f$  pode ser expressa como uma função de sua transformada bidimensional. Reescrevendo a equação 3.9, tem-se que

$$f(x, y) = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} T(u, v) b^{(x,y)}(u, v) \quad (6.26)$$

em que  $N$  foi substituído por  $n$ , pois  $f(x, y)$  agora representa uma subimagem com dimensões  $n \times n$ , enquanto  $b^{(x,y)}(u, v)$  é o núcleo da transformada inversa.

A notação utilizada na equação 6.26 pode ser simplificada para

$$\mathbf{F} = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} T(u, v) \mathbf{H}_{uv} \quad (6.27)$$

em que  $\mathbf{F}$  é uma matriz  $n \times n$  contendo os pixels de  $f$  e  $\mathbf{H}_{uv}$  é definida como

$$\mathbf{H}_{uv} = \begin{bmatrix} h^{(u,v)}(0,0) & h^{(u,v)}(0,1) & \dots & h^{(u,v)}(0, n-1) \\ h^{(u,v)}(1,0) & h^{(u,v)}(1,1) & \dots & h^{(u,v)}(1, n-1) \\ h^{(u,v)}(2,0) & h^{(u,v)}(2,1) & \dots & h^{(u,v)}(2, n-1) \\ \vdots & \vdots & \ddots & \vdots \\ h^{(u,v)}(n-1,0) & h^{(u,v)}(n-1,1) & \dots & h^{(u,v)}(n-1, n-1) \end{bmatrix} \quad (6.28)$$

Portanto, a matriz  $\mathbf{F}$  contendo os pixels da subimagem de entrada é definida como uma combinação linear de  $n^2$  matrizes  $n \times n$ , ou seja,  $\mathbf{H}_{uv}$ , para  $u, v = 0, 1, \dots, n - 1$ . Essas matrizes são as imagens de base da transformada utilizadas no cálculo dos coeficientes de ponderação da expansão em série  $T(u, v)$ .

Uma função de mascaramento dos coeficientes da transformada pode ser definida como

$$m(u, v) = \begin{cases} 0, & \text{se } T(u, v) \text{ satisfizer um determinado critério de truncamento} \\ 1, & \text{caso contrário} \end{cases} \quad (6.29)$$

para  $u, v = 0, 1, \dots, n - 1$ . Uma aproximação de  $\mathbf{F}$  pode ser obtida a partir da expansão após o truncamento

$$\hat{\mathbf{F}} = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} T(u, v) m(u, v) \mathbf{H}_{uv} \quad (6.30)$$

em que  $m(u, v)$  é construído para eliminar as imagens de base que implicarem menor contribuição na soma total na equação 6.28. O erro médio quadrático entre a subimagem  $\mathbf{F}$  e a aproximação  $\hat{\mathbf{F}}$  é, portanto, expresso como

$$e_{MSE} = \|\mathbf{F} - \hat{\mathbf{F}}\|^2 \quad (6.31)$$

As transformadas, nas quais a maior parte da informação é mapeada no menor número possível de coeficientes, permitem melhores aproximações da subimagem e, consequentemente, menores erros de decodificação.

Uma vez realizada a transformação, os coeficientes resultantes podem ser quantizados. A seleção dos coeficientes, baseada na função de mascaramento dada pela equação 6.29, normalmente é realizada pela *codificação por zonas* ou pela *codificação por limiarização*.

Na codificação por zonas, os coeficientes distribuídos nas  $n \times n$  posições da subimagem transformada são divididos em regiões ou zonas. A cada zona, identificada por um índice, corresponde um número de níveis de quantização. Os coeficientes com índice menor que um valor especificado são conservados, enquanto os demais recebem o valor 0. A figura 6.19(a) ilustra uma máscara típica para a codificação por zona.

Na codificação por limiarização, a seleção dos coeficientes é baseada na magnitude de seus valores. Como normalmente os coeficientes da transformada com maior magnitude são os responsáveis pela contribuição mais significativa na qualidade da subimagem decodificada, os coeficientes selecionados são aqueles que excedem um determinado limiar especificado. Há três maneiras básicas para limiarizar uma subimagem transformada, ou seja, construir uma função de mascaramento de limiarização da subimagem, conforme a equação 6.29. Uma forma é utilizar um único limiar global para todas as subimagens. Outra forma é utilizar um limiar diferente para cada subimagem. Uma terceira abordagem é variar o limiar em função da posição de cada coeficiente na subimagem. A figura 6.19(b) ilustra uma máscara típica para a codificação por limiarização.

Após o processo de quantização, deve-se utilizar um método de atribuição de bits aos coeficientes quantizados. Os métodos típicos utilizados nessa etapa são a codificação de Huffman e a codificação aritmética, descritas na seção 6.3.1.

## 6.4 Padronizações de Compressão de Imagens

Vários esforços de padronização dos métodos de compressão de imagens têm sido realizados para facilitar o intercâmbio de imagens comprimidas entre diferentes dispositivos e aplicações. Muitos métodos de compressão descritos na seção 6.3 desempenham um pa-

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

(a)

|   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

(b)

Figura 6.19: Máscaras típicas para codificação (a) por zonas; (b) por limiarização.

pel fundamental no desenvolvimento e na adoção das principais padronizações atuais de compressão de imagens.

O CCITT (do francês *Comité Consultatif International Télégraphique et Téléphonique*), substituído a partir de 1992 pela ITU (*International Telecommunication Union*), e a ISO (*International Standardization Organization*) definiram várias padronizações para compressão de imagens binárias, monocromáticas e coloridas.

#### 6.4.1 Compressão de Imagens Binárias

As padronizações mais largamente utilizadas para compressão de imagens binárias são os padrões dos grupos 3 e 4 do CCITT. Esses padrões foram originalmente desenvolvidos como métodos para codificação e transferência de documentos pela rede telefônica.

A padronização do grupo 3 aplica uma técnica não-adaptativa por código de comprimento de corrida unidimensional em que pelo menos  $k-1$  linhas de cada grupo de  $k$  linhas (para  $k = 2$  e  $k = 4$ ) são codificadas de maneira ótima na abordagem bidimensional. A padronização do grupo 4 é uma versão simplificada da padronização do grupo 3, em que apenas a codificação bidimensional é permitida. Ambas as padronizações utilizam a mesma técnica de codificação bidimensional não-adaptativa.

O grupo JBIG (do inglês, *Joint Bi-level Image Experts Group*), um comitê formado pela ITU e pela ISO, desenvolveu métodos para compressão de texto, imagens meios-tonos e outros conteúdos binários de imagem. Em muitas situações, a padronização JBIG permite um aumento de 20% a 50% na eficiência da compressão em comparação ao padrão do grupo 4.

O JBIG utiliza uma variação da técnica de codificação aritmética, denominada *Q-coder*, a qual se baseia no cálculo das probabilidades de cada bit a partir das linhas anteriores já analisadas da imagem.

#### 6.4.2 Compressão de Imagens Estáticas Monocromáticas e Coloridas

Uma das padronizações mais comuns e completas para imagens estáticas monocromáticas e coloridas é chamada de JPEG (do inglês, *Joint Photographic Experts Group*). Por meio de um conjunto de operações aplicadas sobre a imagem, pode-se regular a taxa de compressão, desde a redução de redundâncias sem perdas até a eliminação de informações que não são facilmente distinguíveis pelo sistema visual humano.

As taxas de compressão dependem da imagem original, entretanto, taxas de 10:1 ou 20:1 podem ser obtidas com alta fidelidade em imagens coloridas. Taxas de 30:1 a 50:1 podem ser alcançadas com baixas a moderadas distorções. Taxas ainda maiores aplicadas em imagens de baixa qualidade também são praticáveis para fins de pré-visualização. Em geral, imagens monocromáticas permitem taxas menos elevadas de compressão com relação às imagens monocromáticas, uma vez que o sistema visual humano é mais sensível a variações de brilho do que a variações de cor.

O JPEG possui quatro modos de operação. No modo *seqüencial*, a imagem é codificada em uma única varredura, ou seja, da esquerda para a direita, de cima para baixo. No modo *progressivo*, a imagem é codificada em múltiplas varreduras, aumentando a qualidade e resolução a cada nova varredura. No modo *reversível*, a imagem é codificada sem perdas de modo a garantir sua exata recuperação. Finalmente, no modo *hierárquico*, a imagem é codificada em múltiplas resoluções, tal que as versões de menor resolução podem ser manipuladas sem a descompressão da imagem com resolução total.

A figura 6.20 ilustra as principais operações da compressão e descompressão JPEG seqüencial. A padronização JPEG 2000, lançado como uma nova versão do JPEG, utiliza a transformada discreta wavelet (DWT, do inglês, *discrete wavelet transform*) em vez da transformada discreta do cosseno (DCT, do inglês, *discrete cosine transform*), ambas descritas no capítulo 3. O JPEG 2000 provê novas funcionalidades para aplicações que utilizam imagens, com desempenho superior na compressão com perdas, transmissão progressiva por resolução, suporte para imagens de dimensões elevadas (acima de 64 Mbytes).

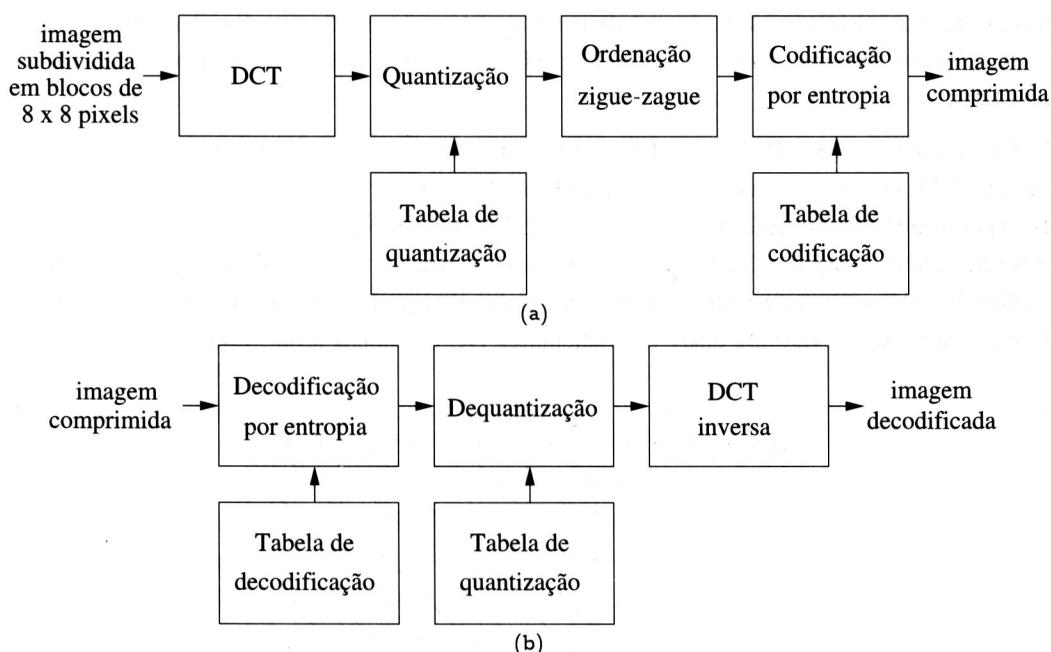


Figura 6.20: Etapas principais da compressão e descompressão JPEG seqüencial.

Para imagens coloridas, o JPEG codifica o espaço de cores de RGB para YCbCr. O apêndice D descreve vários espaços de cores, dentre eles, os espaços RGB e o YCbCr. Os valores das três bandas sofrem um deslocamento de nível de -128. Para imagens mono-

cromáticas, os valores também sofrem um deslocamento de -128. Dessa forma, os valores no intervalo [0, 255] são deslocados para [-128, 127].

A imagem é subdividida em blocos não sobrepostos de  $8 \times 8$  pixels. A transformada discreta do cosseno é aplicada a cada um dos blocos da imagem, conforme a equação

$$F(u, v) = \frac{1}{4} C(u)C(v) \sum_{x=0}^7 \sum_{y=0}^7 f(x, y) \cos \frac{\pi u(2x+1)}{16} \cos \frac{\pi v(2y+1)}{16} \quad (6.32)$$

em que

$$C(u) = C(v) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{para } u, v = 0 \\ 0, & \text{para } u, v > 0 \end{cases} \quad (6.33)$$

Dos valores resultantes,  $F(0,0)$  é chamado de coeficiente DC, enquanto os demais 63 valores são denominados coeficientes AC. Em uma imagem típica, muitos dos coeficientes terão valor zero ou próximo de zero, os quais serão descartados no processo de codificação.

A etapa de quantização aumenta o número de coeficientes com valor zero, eliminando os coeficientes que contribuem pouco para a qualidade da imagem. Os coeficientes resultantes da etapa de DCT são transformados conforme a equação

$$\hat{F}(u, v) = \text{round} \left[ \frac{F(u, v)}{Q(u, v)} \right] \quad (6.34)$$

em que os valores  $Q(u, v)$  formam a tabela de quantização. Cada um dos elementos  $Q(u, v)$  é um número inteiro no intervalo de 1 a 255. O padrão permite até quatro tabelas de quantização.

Os 63 coeficientes AC são reordenados em uma sequência zigue-zague, conforme ilustra a figura 6.21, com o objetivo de facilitar a etapa de codificação por entropia. Os coeficientes de baixa freqüência, os quais têm maior probabilidade de serem diferentes de zero, aparecem antes dos coeficientes de alta freqüência. Já os coeficientes DC, que representam os valores médios dos blocos de  $8 \times 8$  pixels, são codificados por meio de técnicas preditivas em virtude da forte correlação existente entre os coeficientes DC de blocos adjacentes.

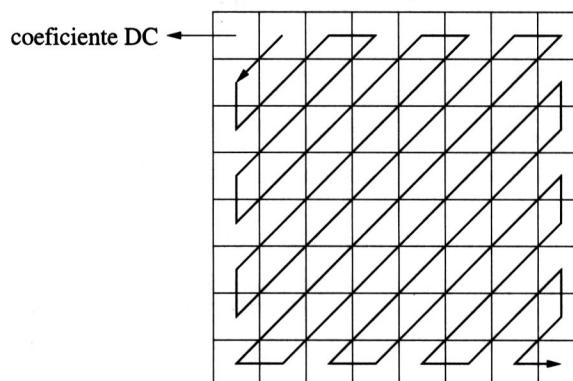


Figura 6.21: Ordenação zigue-zague.

O JPEG especifica dois métodos que podem ser utilizados pelo codificador por entropia, a codificação de Huffman ou a codificação aritmética. Aos símbolos com maior probabilidade de ocorrência são atribuídos códigos binários mais curtos e, aos de menor probabilidade, atribuem-se códigos binários mais longos. Tabelas de Huffman para a codificação seqüencial JPEG são encontradas em Pennebaker e Mitchell (1993).

Na decodificação seqüencial, os códigos binários da imagem comprimida são convertidos em coeficientes DCT pelo decodificador por entropia. Esses coeficientes são dequantizados conforme a equação

$$F(u, v) = \hat{F}(u, v)Q(u, v) \quad (6.35)$$

Os coeficientes dequantizados são mapeados do domínio de freqüência para o domínio espacial por meio da transformada DCT inversa, expressa como

$$f(x, y) = \frac{1}{4} \sum_{u=0}^7 \sum_{v=0}^7 C(u)C(v)F(u, v) \cos \frac{\pi u(2x+1)}{16} \cos \frac{\pi v(2y+1)}{16} \quad (6.36)$$

em que

$$C(u) = C(v) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{para } u, v = 0 \\ 0, & \text{para } u, v > 0 \end{cases} \quad (6.37)$$

Após a aplicação da transformada DCT inversa, os valores decodificados são deslocados pelo valor 128, de modo a reconstruir as intensidades de cinza ou cor.

**Exemplo 6.6 Ilustração da compressão e descompressão JPEG da subimagem 8 × 8 pixels mostrada na figura 6.22.** A imagem original consiste em  $2^8$ , ou seja, 256 possíveis níveis de cinza, de modo que o processo de codificação inicia-se por um deslocamento dos níveis de cinza da subimagem por  $-2^{8-1}$  ou -128 níveis de cinza. O bloco resultante é mostrado na figura 6.22(b).

A transformada DCT, de acordo com a equação 6.32, aplicada ao bloco após o deslocamento dos níveis de cinza produz os resultados mostrados na figura 6.22(c). Apenas uma casa decimal é utilizada na apresentação dos resultados. Após o cálculo da transformada, os 64 coeficientes DCT são quantizados conforme a equação 6.34, realizando-se o arredondamento de cada coeficiente ao seu valor inteiro mais próximo.

Embora nenhuma tabela de quantização  $Q(u, v)$  seja especificada na padronização JPEG, uma tabela típica é mostrada na figura 6.22(d). A partir dessa tabela, os coeficientes do bloco são normalizados e arredondados, resultando na figura 6.22(e). O coeficiente DC, por exemplo, é calculado como  $\hat{F}(0, 0) = \text{round}(F(0, 0)/Q(0, 0)) = \text{round}(226.3/16) = 14$ .

O processo de transformação e quantização dos coeficientes produz um grande número de coeficientes nulos. Com a ordenação dos coeficientes na ordem zig-zague mostrada na figura 6.21, a seqüência unidimensional dos coeficientes é [14 0 -3 -1 -2 -2 0 0 -1 EOB], em que o símbolo EOB denota o final do bloco para indicar à codificação de Huffman que os demais coeficientes em uma seqüência ordenada são nulos.

A decodificação da subimagem comprimida inicia-se com a restauração dos coeficientes da transformada normalizada que levam à seqüência de bits comprimida. Como

a seqüência binária gerada pela técnica de Huffman é decodificada de maneira única, a subimagem resultante dos coeficientes quantizados é a mesma daquela mostrada na figura 6.22(e).

A dequantização dos coeficientes, de acordo com a equação 6.35, gera a subimagem mostrada na figura 6.22(f). O coeficiente DC, por exemplo, é calculado como  $F(0,0) = \hat{F}(0,0)Q(0,0) = (14)(16) = 224$ .

O bloco reconstruído é obtido tomando-se a DCT inversa, conforme equação 6.36, resultando na matriz mostrada na figura 6.22(g), a qual é seguida pelo deslocamento do nível de cinza de cada pixel por +128, resultando na subimagem mostrada na figura 6.22(h). As diferenças entre as subimagens original e reconstruída decorrem dos processos de codificação e decodificação com perdas utilizados na padronização JPEG. Neste exemplo, os erros variam entre -7 (= 132 - 139) e +8 (= 162 - 154).

#### 6.4.3 Compressão de Seqüência de Imagens Monocromáticas e Coloridas

A padronização H.261, recomendada pela ITU para serviços audiovisuais em telecomunicações, inclui métodos de codificação e decodificação de seqüências de imagens para utilização em transmissões por rede a taxas de  $p \times 64$  kbit/s, em que p pode assumir valores entre 1 e 30. As recomendações de H.260 a H.279 referem-se à codificação de vídeo, entretanto, a padronização H.261 e a H.263 são as mais comuns em transmissões por rede.

A padronização H.261 possui muitas semelhanças com o JPEG, em que cada imagem é subdividida em pequenos blocos, nos quais são aplicados a transformada DCT, a quantização e a codificação por comprimento variável. Enquanto o JPEG codifica cada imagem individualmente, o H.261 comprime uma imagem (quadro) inicial ou de referência e estima a diferença (movimento) do quadro atual com base no quadro anterior.

A padronização H.263, baseada na H.261, foi desenvolvida como codificador de vídeo para comunicação em baixas taxas de transmissão, particularmente para a rede telefônica convencional. A exemplo da padronização H.261, a codificação utiliza um processo de predição de movimento entre quadros adjacentes em conjunto com a codificação DCT do erro de predição. O desempenho da compressão é melhorado com a identificação de redundâncias temporais.

O MPEG (do inglês, *Moving Picture Experts Group*), estabelecido pela ISO, corresponde a um conjunto de padrões de codificação digital de vídeo e áudio para operação em diferentes taxas de dados.

O padrão inicial MPEG-1 foi desenvolvido para aplicações de vídeos a taxas de 1.5 Mbits/s, os quais podem ser armazenados em meios digitais como CD-ROM (do inglês, *compact disc - read only memory*). O MPEG-1 inclui três padrões de áudio e muitos sistemas de vídeo utilizam a camada de áudio 1 ou 2 do MPEG-1. A camada de áudio 3 do MPEG-1, conhecida como MP3, tem sido amplamente utilizada para áudio na Internet.

O MPEG-2 foi desenvolvido para permitir taxas de dados maiores do que as do MPEG-1 e, dessa forma, maior qualidade. O MPEG-2 é o padrão utilizado em aparelhos de DVD (do inglês, *digital versatile disc*) e no sistema de TV digital norte-americana. O MPEG-3 foi abandonado em 1993, já que as funções planejadas foram incluídas no MPEG-2.

|   |       |       |      |      |      |      |      |                                  |     |     |     |     |     |     |     |
|---|-------|-------|------|------|------|------|------|----------------------------------|-----|-----|-----|-----|-----|-----|-----|
| 133                                       | 132   | 147   | 151  | 158  | 158  | 159  | 160  | 5                                | 4   | 19  | 23  | 30  | 30  | 31  | 32  |
| 142                                       | 138   | 149   | 152  | 157  | 154  | 154  | 152  | 14                               | 10  | 21  | 24  | 29  | 26  | 26  | 24  |
| 153                                       | 149   | 162   | 161  | 159  | 152  | 152  | 151  | 25                               | 21  | 34  | 33  | 31  | 24  | 24  | 23  |
| 162                                       | 156   | 157   | 158  | 157  | 157  | 158  | 157  | 34                               | 28  | 29  | 30  | 29  | 29  | 30  | 29  |
| 159                                       | 160   | 163   | 161  | 160  | 154  | 153  | 152  | 31                               | 32  | 35  | 33  | 32  | 26  | 25  | 24  |
| 165                                       | 162   | 162   | 160  | 162  | 156  | 154  | 158  | 37                               | 34  | 34  | 32  | 34  | 28  | 26  | 30  |
| 160                                       | 164   | 163   | 160  | 161  | 158  | 152  | 154  | 32                               | 36  | 35  | 32  | 33  | 30  | 24  | 26  |
| 161                                       | 160   | 166   | 164  | 167  | 162  | 157  | 156  | 33                               | 32  | 38  | 36  | 39  | 34  | 29  | 28  |
| (a) bloco $8 \times 8$ da imagem original |       |       |      |      |      |      |      | (b) bloco após subtração por 128 |     |     |     |     |     |     |     |
| 226.3                                     | -2.2  | -15.9 | -1.8 | 5.3  | 1.1  | 4.8  | 6.6  | 16                               | 11  | 10  | 16  | 24  | 40  | 51  | 61  |
| -31.2                                     | -27.7 | -4.8  | -0.7 | 1.9  | 6.0  | 1.5  | 2.6  | 12                               | 12  | 14  | 19  | 26  | 58  | 60  | 55  |
| -8.6                                      | -18.7 | -7.6  | -0.8 | -0.7 | -1.4 | 2.1  | 1.3  | 14                               | 13  | 16  | 24  | 40  | 57  | 69  | 56  |
| -4.4                                      | -6.6  | 2.7   | -0.2 | 0.0  | -3.3 | -0.8 | -0.7 | 14                               | 17  | 22  | 29  | 51  | 87  | 80  | 62  |
| 3.3                                       | -6.9  | 0.2   | 1.1  | -1.1 | 1.2  | -0.2 | -0.2 | 18                               | 22  | 37  | 56  | 68  | 109 | 103 | 77  |
| 2.1                                       | -1.5  | 3.9   | -0.1 | -0.9 | -0.6 | 0.1  | -2.0 | 24                               | 35  | 55  | 64  | 81  | 104 | 113 | 92  |
| 5.6                                       | 0.4   | -1.3  | -2.3 | 2.3  | 1.1  | 1.3  | 1.9  | 49                               | 64  | 78  | 87  | 103 | 121 | 120 | 101 |
| -1.8                                      | 4.3   | -6.4  | -4.6 | -1.8 | -0.5 | 0.1  | 0.6  | 72                               | 92  | 95  | 98  | 112 | 100 | 103 | 99  |
| (c) DCT do bloco                          |       |       |      |      |      |      |      | (d) tabela de quantização        |     |     |     |     |     |     |     |
| 14  | 0     | -2    | 0    | 0    | 0    | 0    | 0    | 224                              | 0   | -20 | 0   | 0   | 0   | 0   | 0   |
| -3  | -2    | 0     | 0    | 0    | 0    | 0    | 0    | -36                              | -24 | 0   | 0   | 0   | 0   | 0   | 0   |
| -1  | -1    | 0     | 0    | 0    | 0    | 0    | 0    | -14                              | -13 | 0   | 0   | 0   | 0   | 0   | 0   |
| 0   | 0     | 0     | 0    | 0    | 0    | 0    | 0    | 0                                | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0   | 0     | 0     | 0    | 0    | 0    | 0    | 0    | 0                                | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0   | 0     | 0     | 0    | 0    | 0    | 0    | 0    | 0                                | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0   | 0     | 0     | 0    | 0    | 0    | 0    | 0    | 0                                | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| 0   | 0     | 0     | 0    | 0    | 0    | 0    | 0    | 0                                | 0   | 0   | 0   | 0   | 0   | 0   | 0   |
| (e) bloco DCT quantizado                  |       |       |      |      |      |      |      | (f) bloco dequantizado           |     |     |     |     |     |     |     |
| 7.4                                       | 10.7  | 15.8  | 21.0 | 24.4 | 25.7 | 25.5 | 24.9 | 135                              | 139 | 144 | 149 | 152 | 154 | 154 | 153 |
| 12.3                                      | 15.2  | 19.6  | 23.8 | 26.2 | 26.5 | 25.5 | 24.6 | 140                              | 143 | 148 | 152 | 154 | 155 | 154 | 153 |
| 20.0                                      | 22.3  | 25.6  | 28.2 | 29.0 | 27.9 | 25.7 | 24.1 | 148                              | 150 | 154 | 156 | 157 | 156 | 154 | 152 |
| 27.5                                      | 29.2  | 31.4  | 32.6 | 31.9 | 29.3 | 26.1 | 23.9 | 156                              | 157 | 159 | 161 | 160 | 157 | 154 | 152 |
| 32.3                                      | 33.6  | 35.1  | 35.6 | 33.9 | 30.5 | 26.7 | 24.1 | 160                              | 162 | 163 | 164 | 162 | 159 | 155 | 152 |
| 33.7                                      | 34.9  | 36.3  | 36.6 | 34.8 | 31.2 | 27.3 | 24.7 | 162                              | 163 | 164 | 165 | 163 | 159 | 155 | 153 |
| 32.7                                      | 34.1  | 35.7  | 36.3 | 34.8 | 31.6 | 27.8 | 25.4 | 161                              | 162 | 164 | 164 | 163 | 160 | 156 | 153 |
| 31.5                                      | 32.9  | 34.9  | 35.7 | 34.6 | 31.7 | 28.2 | 25.8 | 160                              | 161 | 163 | 164 | 163 | 160 | 156 | 154 |
| (g) DCT inversa do bloco                  |       |       |      |      |      |      |      | (h) bloco decodificado           |     |     |     |     |     |     |     |

Figura 6.22: Exemplo de compressão e descompressão JPEG seqüencial.

O MPEG-4 permite a transmissão de vídeo com qualidade comparada ao MPEG-1, porém, com uma taxa de bits muito menor. O MPEG-4 provê objetos dinâmicos que podem ser transmitidos separadamente e combinados para formar o quadro do vídeo, possibilidade de interferência do usuário para composição de cenas e qualidade audiovisual superior aos outros padrões de codificação a taxas de dados comparáveis.

O MPEG-7 é uma norma para descrição de conteúdos digitais de modo a permitir acesso eficiente aos vários tipos de informações, tais como imagens estáticas, vídeo, áudio, gráficos, textos. O MPEG-7 especifica um conjunto de descritores e seus relacionamentos, bem como formas para o usuário definir suas próprias estruturas. A definição de novos descritores ou esquemas de descrição é feita a partir de uma linguagem específica. Diversas aplicações e

domínios de conhecimento podem se beneficiar do padrão MPEG-7, tais como bibliotecas digitais, educação, medicina, entretenimento e sistemas de informações geográficas.

O MPEG-21 define um conjunto de recursos para a distribuição de conteúdos multimídia, incluindo a definição da sintaxe e semântica de seus componentes. Esses recursos oferecem melhores mecanismos para uso dos conteúdos sobre uma variedade de redes e dispositivos.

## 6.5 Notas Bibliográficas

Material adicional sobre técnicas de compressão de imagens pode ser consultado em Hirschberg e Lelewer (1990), Wintz (1972), Delp e Mitchell (1979), Bentley et al. (1986), Habibi (1974), Farrelle (1990), Jain (1981), Netravali e Haskell (1995), Einarsson (1991), Clarke (1986), Rabbani e Jones (1991) e Gonzalez e Woods (2000).

Conceitos sobre teoria da informação são descritos em Cover e Thomas (1991), Gray (1990), Pierce (1980) e Abramson (1963).

Técnicas de quantização são descritas por Kretz (1975), Kretz (1964), Wood (1969), Gersho e Gray (1992) e Goldberg et al. (1986). Material sobre a técnica DPCM é encontrado em Daut e Zhao (1990) e Habibi (1971). A modulação delta é descrita por Jager (1952) e Schindler (1974).

A técnica de codificação de Huffman é descrita por Knuth (1985) e Vitter (1987). Material adicional sobre codificação aritmética pode ser consultado em Gonzalez et al. (1990) e Witten e Cleary (1997).

Material sobre padronização para compressão de imagens e vídeo pode ser consultado em Bhaskaran e Konstantinides (1995), Aravind et al. (1995), Wallace (1991), Hunter e Robinson (1980) e LeGall (1991).

Análises comparativas entre métodos de compressão são encontradas em Lovewell e Basart (1988) e Heer e Reinfelder (1990).

## 6.6 Problemas

1. Suponha a imagem monocromática mostrada na figura 6.23.

|    |    |    |    |     |     |     |     |
|----|----|----|----|-----|-----|-----|-----|
| 21 | 21 | 21 | 95 | 169 | 243 | 243 | 243 |
| 21 | 21 | 21 | 95 | 169 | 243 | 243 | 243 |
| 21 | 21 | 21 | 95 | 169 | 243 | 243 | 243 |
| 21 | 21 | 21 | 95 | 169 | 243 | 243 | 243 |

Figura 6.23: Imagem monocromática.

- a) Calcule a entropia da imagem.
- b) Calcule o código de Huffman para cada nível de cinza da imagem.
2. Descreva os principais tipos de redundância que podem ser explorados pelos métodos de compressão de imagens.

3. Descreva dois métodos de compressão de imagens com perdas.
4. Qual a vantagem da ordenação zigue-zague dos coeficientes DCT na padronização JPEG?
5. Represente a imagem binária mostrada na figura 6.24 pelo código de comprimento de corridas, iniciando-se com o comprimento das corridas de valor 1.

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |

Figura 6.24: Imagem binária.

6. Com base na distribuição de probabilidade apresentada na tabela 6.12, aplique a decodificação aritmética para a mensagem 0.233355.

Tabela 6.12: Distribuição de probabilidade para a decodificação aritmética.

| Símbolo | Probabilidade |
|---------|---------------|
| a       | 0.2           |
| b       | 0.3           |
| c       | 0.1           |
| d       | 0.2           |
| e       | 0.1           |
| f       | 0.1           |

7. Explique o princípio das técnicas preditivas de compressão de imagens. Descreva a principal diferença entre técnicas preditivas sem perdas e com perdas.
8. O código de Gray é uma codificação binária em que duas palavras consecutivas diferem em apenas um bit.
  - a) Construa o código de Gray de 4 bits.
  - b) Descreva um procedimento geral para geração do código de Gray a partir do código binário.
9. Explique as vantagens e desvantagens em se utilizar blocos de tamanhos diferentes de  $8 \times 8$  pixels no cálculo da transformada DCT na padronização JPEG.
10. Considere uma fonte que emita símbolos de um alfabeto  $A = \{s_1, s_2, s_3, s_4\}$  com probabilidades  $p(s_1) = 0.3$ ,  $p(s_2) = 0.0$ ,  $p(s_3) = 0.4$  e  $p(s_4) = 0.3$ . A entropia dessa fonte é aproximadamente igual a 1.571 bits/símbolo. Encontre os códigos de Huffman para essa fonte e calcule o seu comprimento médio. Compare esse valor com a entropia.

11. Considere uma fonte com alfabeto  $\Omega = \{a, b, c\}$ . A partir da seqüência 1, 2, 2, 0, 2, 3, 0, 1, 9, 10, 5, 9, construa o dicionário e decodifique essa seqüência por meio do algoritmo LZW. tem que ser menor do que 9
12. Quantos códigos de Huffman distintos existem para uma fonte de três símbolos?
13. Escreva um programa para codificar uma imagem binária pelo código do comprimento de corrida. Decodifique a imagem comprimida resultante.
14. Dada uma fonte de cinco símbolos com probabilidades {0.55, 0.15, 0.15, 0.10, 0.05}, obtenha duas codificações de Huffman diferentes. Calcule o comprimento médio  $\bar{L}$  de bits para cada um dos códigos.
15. Considere uma fonte com  $m$  símbolos equiprováveis. Por que o código mais eficiente deve ter palavras de comprimento fixo?

0 1 2 3 4 5 6 7 8 9  
 a b c bc cc ra ac cb bra ab bab

## Referências Bibliográficas

- Abramson, N. (1963). *Information Theory and Coding*, McGraw-Hill, Nova York, NY, Estados Unidos.
- Aravind, R., Cash, G. L., Hang, H.-M., Haskell, B. G. e Puri, A. (1995). *Image and Video Coding Standards*, Academic Press, Inc., Duluth, MN, Estados Unidos.
- Bentley, J. L., Sleator, D. D., Tarjan, R. E. e Wei, V. K. (1986). A Locally Adaptive Data Compression Scheme, *Communications of the ACM* 29(4): 320–330.
- Bhaskaran, V. e Konstantinides, K. (1995). *Image and Video Compression Standards*, Kluwer Academic Publishers.
- Clarke, R. J. (1986). *Transform Coding of Images*, Academic Press, San Diego, CA, Estados Unidos.
- Cover, T. M. e Thomas, J. A. (1991). *Elements of Information Theory*, John Wiley and Sons, Inc., Nova York, NY, Estados Unidos.
- Daut, D. G. e Zhao, D. (1990). Improved DPCM Algorithm for Image Data Compression, *Image Processing Algorithms and Techniques*, Bellingham, WA, Estados Unidos, pp. 199–210.
- Delp, E. J. e Mitchell, O. R. (1979). Image Coding Using Block Truncation Coding, *IEEE Transactions on Communications* 27(9): 1335–1342.
- Einarsson, G. (1991). An Improved Implementation of Predictive Coding Compression, *IEEE Transactions on Communications* 39: 169–171.
- Fano, R. M. (1949). *Transmission of Information*, MIT Press, Cambridge, MA, Estados Unidos.
- Farrelle, M. (1990). *Recursive Block Coding for Image Data Compression*, Springer-Verlag, Nova York, NY, Estados Unidos.
- Gersho, A. e Gray, R. M. (1992). *Vector Quantization and Signal Compression*, Kluwer Academic Press.
- Goldberg, M., Boucher, P. R. e Shlien, S. (1986). Image Compression Using Adaptive Vector Quantization, *IEEE Transactions on Communications* 34(2): 180–187.
- Gonzalez, C. A., Anderson, K. L. e Pennebaker, W. B. (1990). DCT Based Video Compression Using Arithmetic Coding, *Image Processing Algorithms and Techniques*, Bellingham, WA, Estados Unidos, pp. 305–311.
- Gonzalez, R. C. e Woods, R. E. (2000). *Processamento de Imagens Digitais*, Edgard Blücher, São Paulo, SP.

- Gray, R. (1990). *Entropy and Information Theory*, Springer-Verlag, Nova York, NY, Estados Unidos.
- Guazzo, M. (1980). A General Minimum-Redundancy Source-Coding Algorithm, *IEEE Transactions on Information Theory* 26(1): 15–25.
- Habibi, A. (1971). Comparison of  $n^{\text{th}}$ -Order DPCM Encoder with Linear Transformations and Block Quantization Techniques, *IEEE Transactions on Communication Technology* 19(6): 948–956.
- Habibi, A. (1974). Hybrid coding of pictorial data, *IEEE Transactions on Communication Technology* 22(4): 614–624.
- Heer, V. K. e Reinfelder, H. E. (1990). A Comparison of Reversible Methods for Data Compression, *Proceedings of the SPIE: Medical Imaging IV* 1233: 354–365.
- Hirschberg, D. S. e Lelewer, D. A. (1990). Efficient Decoding of Prefix Codes, *Communications of the ACM* 33(4): 449–459.
- Huffman, D. A. (1952). A Method for the Construction of Minimum-Redundancy Codes, *Proceedings of the Institute of Radio Engineers* 40(9): 1098–1101.
- Hunter, R. e Robinson, A. H. (1980). International Digital Facsimile Coding Standards, *Proceedings of the IEEE* 68(7): 854–867.
- Jager, F. D. (1952). Delta Modulation - A Method of PCM Transmission Using the One Unit Code, *Philips Research Report* 7: 442–466.
- Jain, A. K. (1981). Image Data Compression: A Review, *Proceedings of the IEEE* 69(3).
- Knuth, D. E. (1985). Dynamic Huffman coding, *Journal of Algorithms* 6(2): 163–180.
- Kretz, F. (1964). Quantizing for Minimum Distortion, *IEEE Transactions on Information Theory* 10(4): 384–385.
- Kretz, F. (1975). Subjectively Optimal Quantization of Pictures, *IEEE Transactions on Communications* 23(11): 1288–1292.
- LeGall, D. J. (1991). MPEG: A Video Compression Standard for Multimedia Applications, *Communications of the ACM* 34(4): 47–58.
- Lovewell, B. K. e Basart, J. P. (1988). Survey of Image Compression Techniques, *Review of Progress in Quantitative Nondestructive Evaluation* 7A: 731–738.
- Netravali, A. N. e Haskell, B. G. (1995). *Digital Pictures: Representation and Compression*, Plenum Press, Nova York, NY, Estados Unidos.
- Pennebaker, W. B. e Mitchell, J. L. (1993). *JPEG Still Image Data Compression Standard*, Van Nostrand Reinhold, Nova York, NY, Estados Unidos.
- Pierce, J. R. (1980). *An Introduction to Information Theory: Symbols, Signals and Noise*, Dover Publications Inc., Nova York, NY, Estados Unidos.
- Rabbani, M. e Jones, P. W. (1991). *Digital Image Compression Techniques*, Society of Photo-Optical Instrumentation Engineers, Bellingham, WA, Estados Unidos.
- Rissanen, J. (1976). Generalized Kraft Inequality and Arithmetic Coding, *IBM Journal of Research and Development* 20(3): 198–203.
- Rissanen, J. e Langdon, G. G. (1979). Arithmetic Coding, *IBM Journal of Research and Development* 23(2): 149–162.
- Rubin, F. (1979). Arithmetic Stream Coding Using Fixed Precision Registers, *IEEE Transactions on Information Theory* 25(6): 672–675.
- Schindler, H. (1974). Linear, Nonlinear, and Adaptive Delta Modulation, *IEEE Transactions on Communications* 22(11): 1807–1823.
- Shannon, C. E. (1948). A Mathematical Theory of Communication, *Bell System Technical Journal* 27: 379–423.

- Vitter, J. S. (1987). Design and Analysis of Dynamic Huffman Codes, *Journal of the ACM* 34(4): 825–845.
- Wallace, G. K. (1991). The JPEG Still Picture Compression Standard, *Communications of the ACM* 34(4): 30–44.
- Welch, T. A. (1984). A Technique for High-Performance Data Compression, *Computer* 17(6): 8–19.
- Wintz, P. A. (1972). Transform Picture Coding, *Proceedings of the IEEE* 60(7): 809–820.
- Witten, I. H. e Cleary, R. M. N. J. G. (1997). Arithmetic Coding for Data Compression, *Communications of the ACM* 30(6): 520–540.
- Wood, R. (1969). On Optimum Quantization, *IEEE Transactions on Information Theory* 15(2): 248–252.
- Ziv, J. e Lempel, A. (1977). A Universal Algorithm for Sequential Data Compression, *IEEE Transactions on Information Theory* 23(3): 337–342.
- Ziv, J. e Lempel, A. (1978). Compression of Individual Sequences Via Variable-Rate Coding, *IEEE Transactions on Information Theory* 24(5): 530–536.