INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

FORMATOS de IMAGEM

por

João Manuel Brisson Lopes

Departamento de Engenharia Informática

texto elaborado para a disciplina de

Computação Gráfica

Licenciatura em Engenharia Informática e de Computadores

publicado em Janeiro de 2003

reeditado em Dezembro de 2008, Abril 2013

Este texto, elaborado no contexto da disciplina de Computação Gráfica da Licenciatura em Engenharia Informática e de Computadores do Instituto Superior Técnico, foi originalmente concebido para fazer parte de um conjunto de textos sobre Computação Gráfica, apresentando-se agora como um texto independente.

Contacto do autor: brisson@ist.utl.pt

Formatos de Imagem

1 Introdução

O maior número de resultados produzidos por aplicações de Computação Gráfica é constituído por imagens que, ou são transmitidas para um destinatário que as observa à medida que são geradas, ou são armazenadas. Em qualquer dos casos, não basta transmitir ou armazenar o conteúdo das imagens por si só. É também necessário transmitir ou guardar a informação que descreve as imagens, tal como as suas dimensões, emprego de cores, etc. Toda esta informação tem que se encontrar devidamente estruturada para que as aplicações que apresentem as imagens possam interpretar correctamente a informação transmitida ou armazenada. Por outras palavras, é necessário convencionar ou normalizar a estrutura desta informação, ou seja, o formato das imagens.

As primeiras imagens produzidas por aplicações de Computação Gráfica eram do tipo vectorial. Eram relativamente fáceis de armazenar pois eram imagens a preto e branco de que bastava guardar os comandos vectoriais correspondentes às linhas e caracteres que as compunham[1]. No entanto, logo se colocou o problema de como representar externamente os comandos, as coordenadas dos pontos e as cadeias de caracteres que as integravam, tendo surgido dois tipos de solução. Enquanto a Hewlett-Packard optou por uma descrição textual (em ASCII) dos comandos através do formato HP-GL[2], que era legível por qualquer pessoa, a Tektronix optou por um formato binário, onde tanto os comandos como os valores das coordenadas se encontravam codificados em forma binária. Este formato binário, o formato Tek 4010/14, pretendia reduzir ao mínimo o espaço ocupado pelas imagens a transmitir ou armazenar.

Com o aumento da funcionalidade dos equipamentos gráficos de saída houve que modificar estes dois formatos de forma a poderem suportar unidades traçadoras[3] com mais do que uma caneta[4] e terminais gráficos a cores. Por outro lado, a generalização de

[1] Quatro comandos (MOVETO, LINETO, STRING e CLEAR) eram suficientes para este tipo de imagens.
[2] Hewlett-Packard Graphics Language, que cresceu e foi sucessivamente enriquecida, sendo hoje empregue como linguagem de descrição de páginas.
[3] Plotters.
[4] O número de canetas dependia da unidade traçadora.

unidades gráficas de saída do tipo quadrícula (*raster*) veio alterar profundamente a situação e, em consequência, assistiu-se à multiplicação de novos formatos e ao seu progressivo aperfeiçoamento.

Neste capítulo, apresentaremos alguns dos formatos de imagem de uso mais generalizado, além de outros, como os formatos PBM e de imagens de aplicações em Medicina, de interesse mais didáctico ou informativo. Dada a sua preponderância, consideraremos apenas formatos do tipo de quadrícula.

Apresentaremos, inicialmente, os conceitos mais relevantes para o tema tratado, como os conceitos de mapa de cores, entrelaçamento, cores reais e compressão de imagem. Nas secções seguintes, serão sucessivamente analisados o formato PBM, que, devido à sua simplicidade, constitui um óptimo ponto de partida sob o ponto de vista didáctico, o formato DIB, bastante semelhante ao formato PBM, e o formato GIF que, durante muitos anos, e mesmo actualmente, é um formato de imagem extremamente popular, embora apresente limitações ao seu emprego a imagens de qualidade fotográfica. De seguida, apresentaremos o formato PNG, criado para ultrapassar as limitações do formato GIF, e o formato JFIF (JPEG), destinado a imagens de qualidade fotográfica. O formato GE Sygna, apresentado por último, destina-se a dar uma ideia dos requisitos a suportar pelos formatos de imagens em aplicações à Medicina. Estes requisitos derivam da necessidade de acompanhar tais imagens de informação constituindo numa verdadeira ficha do paciente e das condições em que os exames foram realizados.

Finalmente, apresentaremos um resumo das principais características dos formatos descritos e exemplos comparativos do espaço ocupado por imagens nesses formatos.

1.1 Imagens de Quadrícula

Com o advento dos terminais gráficos de quadrícula deixou de ser possível recuperar o conteúdo das imagens em forma vectorial. A representação de imagens passou a ter que armazenar a informação de todos os pontos da quadrícula, os píxeis das imagens, a sua cor e o número de linhas e de colunas da quadrícula, além de outra informação como a referente às dimensões reais dos píxeis, pois a relação entre a sua altura e largura pode variar de unidade para unidade gráfica e causar distorções. Entre outra informação que, mais tarde, foi necessário incluir, encontra-se a correcção gama aplicada às imagens, por forma a evitar que uma mesma imagem apresentada em diversos equipamentos por visualizadores da World Wide Web possa aparentar cores escurecidas num equipamento e cores berrantes num outro.

Nos primórdios do emprego de equipamentos gráficos de quadrícula surgiram muitos formatos de imagem. Uns desapareceram entretanto, outros continuam ainda hoje a ser empregues. Os mais simples destes formatos limitam-se a indicar a largura e a altura da imagem (em píxeis), o tipo de imagem (a cor ou em tons de cinzento) e, depois, os valores das intensidades RGB de cada píxel para imagens a cor, ou o valor da intensidade de luz de cada píxel para imagens em tons de cinzento, segundo uma ordem convencionada, a convenção de varrimento dos píxeis da imagem.

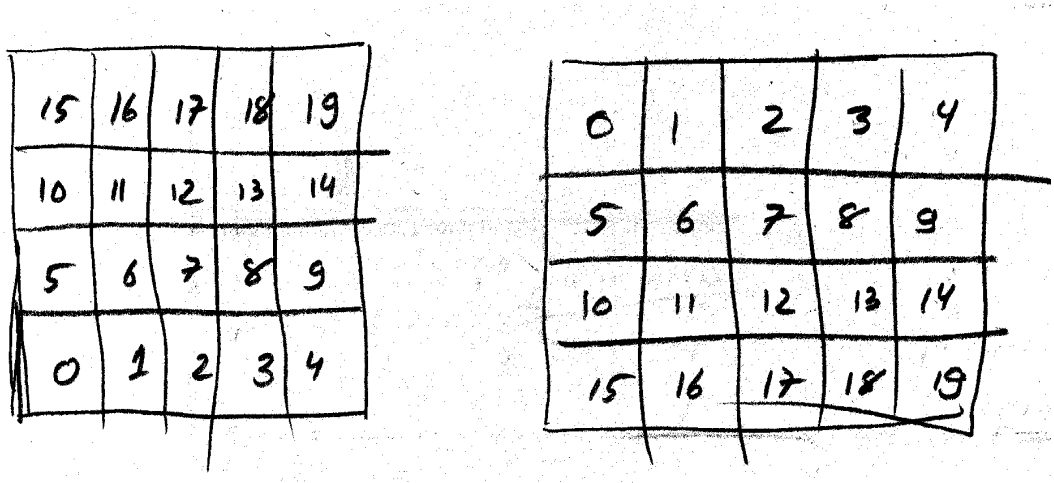


Figura 1.1 – Ordenamento dos píxeis de uma imagem por varrimento ascendente das linhas (à esquerda) e por varrimento descendente (à direita).

É prática corrente armazenar a informação sobre os píxeis de uma imagem varrendo os píxeis de cada linha da esquerda para a direita. As linhas são por sua vez armazenadas consecutivamente, uma linha atrás da precedente, seguindo uma ordem que pode ser ascendente (da linha inferior da imagem para a linha superior) ou, o que é mais comum, descendente (da linha superior para a linha inferior).

1.2 Mapa de Cores

Inicialmente, os terminais gráficos de quadrícula podiam apenas apresentar um número muito reduzido de cores, primeiro 16 e, mais tarde, 256. Para uma imagem a cores de dimensão típica dessa altura, constituída por 360 linhas e 270 colunas, num total de 97.200 píxeis, era necessário armazenar o triplo do número de píxeis (291.600) em unidades de informação[5]. A transmissão de uma tal imagem através de uma linha de grande velocidade (19200 bps) de então[6] demorava cerca de 137 segundos, no melhor dos casos.

A introdução de mapas de cores permitiu reduzir este tempo. Um mapa de cores é uma estrutura que declara as componentes RGB das cores empregues pela imagem a que se encontra associada, atribuindo a cada cor um índice único. Cada cor declarada num mapa de cores ocupa 3 unidades de informação, ou seja, para 256 cores, o mapa de cores ocupará 768 unidades de informação. Se a informação de cada píxel for constituída pelo índice da respectiva cor, cada píxel ocupará apenas uma unidade de informação e não 3. Para a imagem das dimensões que estamos a considerar, passa a ser necessário armazenar 97.968 (768+97.200) unidades de informação. O emprego do mapa de cores permite assim reduzir o espaço ocupado pela imagem a 33,6% do espaço originalmente necessário. O tempo de transmissão da imagem será igualmente reduzido na mesma proporção. Note-se que, embora a imagem seja de pequena dimensão, este valor de 33,6% está já muito próximo do valor limite que é de 33,3%.

[5] Uma unidade de informação corresponde a 4 bits se a imagem empregar 16 cores e a 8 bits (1 byte) se empregar 256 cores.
[6] Linha de transmissão em série, com um único bit de paragem e sem bit de paridade.

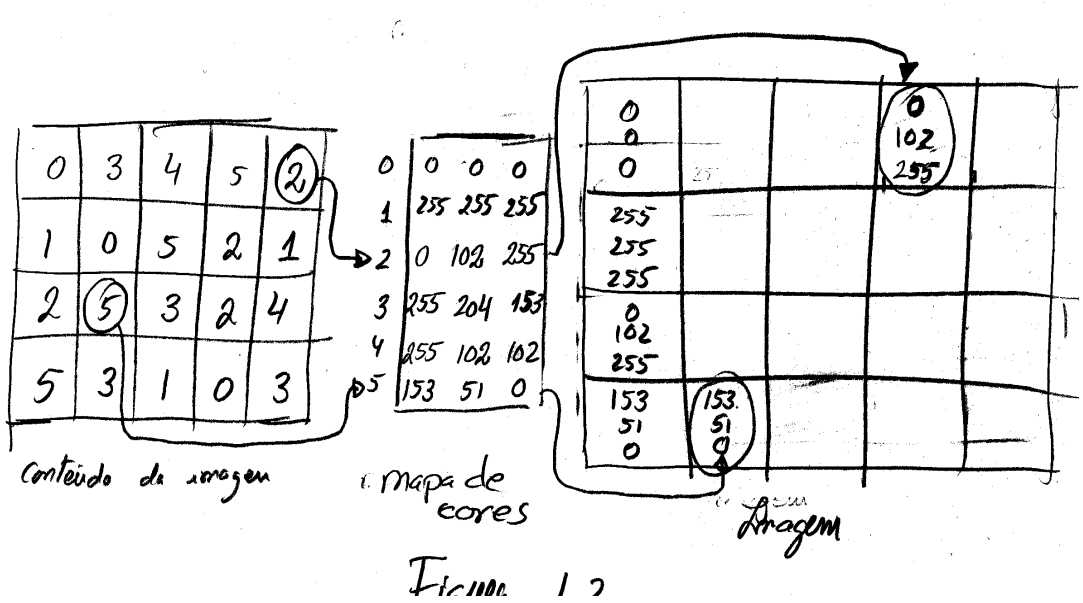


Figura 1.2 – Mapa de cores. Por cada píxel, a imagem contém um índice que é transformado nas componentes de cor na apresentação da imagem num ecrã.

1.3 Entrelaçamento

Mesmo com a redução de tamanho permitida pelo emprego de um mapa de cores, a imagem do exemplo anterior levaria ainda cerca de 45,9 segundos a ser transmitida. O utilizador teria que esperar todo este tempo até poder observar a imagem e ficar com uma ideia do seu conteúdo. Ora acontece que muita da informação constante de uma imagem é redundante, isto é, não é necessário visualizar todos os píxeis de uma imagem para formar uma ideia do seu conteúdo, principalmente quando se procura uma imagem. Foi exactamente para permitir a observação do conteúdo de uma imagem sem dispor de todos os seus píxeis que foi desenvolvida a técnica do entrelaçamento a partir de uma técnica semelhante empregue na transmissão de imagens de televisão.

A técnica básica de transmissão parcial e progressiva de imagens por entrelaçamento consiste em reordenar as linhas das imagens, organizando-as em vários grupos. Cada grupo contém parte das linhas da imagem total e, se uma linha for atribuída a um grupo, essa linha não fará parte de qualquer outro grupo. A transmissão da imagem grupo a grupo permite que o utilizador comece a formar uma ideia da imagem após apenas algumas linhas terem sido transmitidas. O utilizador começa por ver uma imagem nos seus traços gerais, com poucas linhas e sem detalhes, e, à medida que vão sendo apresentados novos grupos de linhas, verificará um aumento progressivo da qualidade e do número de detalhes[7].

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

[7] Esta característica pode ainda hoje ser observada durante o carregamento de imagens transmitidas pela WWW quando o fluxo de dados é baixo.

processa assim 1/8 da imagem, a terceira 1/4 e a quarta passagem a metade restante das linhas.

Para o exemplo anteriormente apresentado de uma imagem de 360×270 píxeis e com 256 cores, todas as linhas do primeiro grupo serão visíveis ao fim de 6,1 segundos[8], as do segundo grupo ao fim de 11,7 segundos e as do terceiro grupo ao fim de 23,1 segundos. A imagem ficará completa com o quarto grupo ao fim de 45,9 segundos.

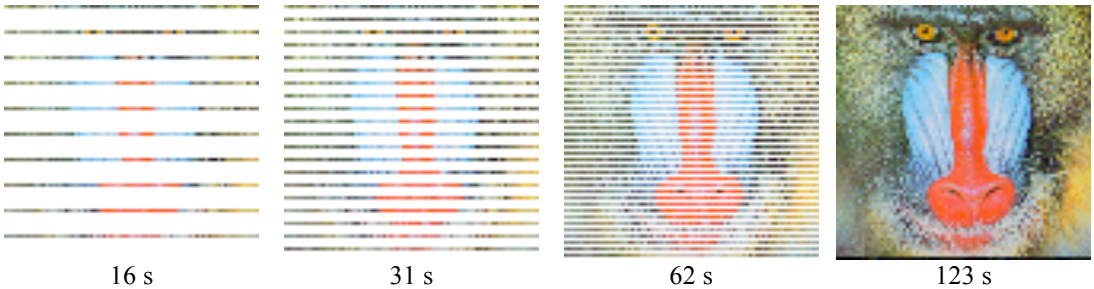
A figura 1.3 apresenta as imagens parciais resultantes da transmissão de uma imagem com o entrelaçamento do formato de imagem GIF.

Um outro tipo de entrelaçamento processa píxeis individuais em lugar de processar linhas. Os píxeis a transmitir em cada passagem são determinados por um padrão de n×n píxeis de que são feitas tantas cópias quanto as necessárias para cobrir toda a imagem, estando as cópias justapostas. O algoritmo de entrelaçamento Adam7, empregue pelo formato de imagem PNG, define um padrão de 8×8 píxeis

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

que aplica a toda a imagem, começando pelo canto superior esquerdo. Se, por exemplo, a imagem a transmitir apresentar a dimensão de 16×16 píxeis, cada uma das 7 passagens transmitirá novas linhas cujo número e comprimento variarão com a ordem da passagem, tal como a tabela 1.1 apresenta. A figura 1.4 apresenta o aspecto da imagem da figura 1.3 após algumas das passagens de entrelaçamento realizadas segundo o algoritmo de entrelaçamento Adam7.

Retomando o exemplo anterior, as 7 passagens do algoritmo de entrelaçamento Adam7 encontrar-se-ão completas ao fim de 1,1s, 1,8s, 3,2s, 6,1s, 11,8s, 23,1s e 45,9s, respectivamente.



16 s 31 s 62 s 123 s

Figura 1.3 – Imagens parciais e respectivos tempos de apresentação de uma imagem de 512×512 píxeis pelo algoritmo de entrelaçamento do formato GIF. Os tempos são contados em relação ao início da apresentação da imagem.

[8] Não esquecer que o mapa de cores tem que ser transmitido antes das linhas.

Passagem nº	Nº de linhas transmitidas	Comprimento das linhas	Bytes transmitidos
1	2	2	4
2	2	2	4
3	2	4	8
4	4	4	16
5	4	8	32
6	8	8	64
7	8	16	128
		Total	256

Tabela 1.1 – Número e comprimento das linhas transmitidas de uma imagem de 16×16 píxeis pelo algoritmo de entrelaçamento Adam7 (formato PNG).

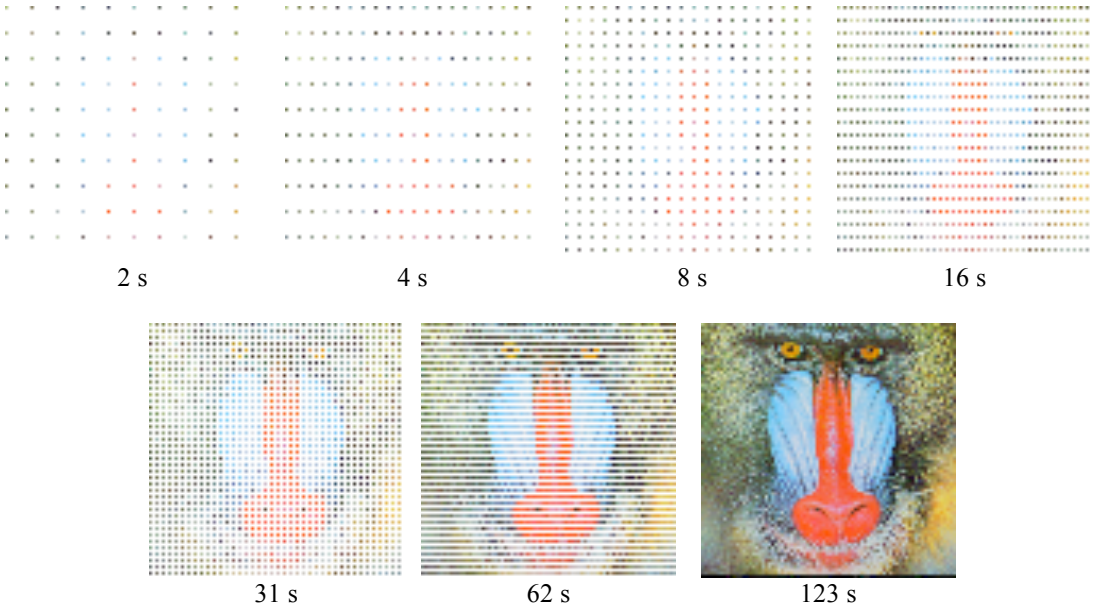


Figura 1.4 – Imagens parciais e respectivos tempos de apresentação de uma imagem de 512×512 píxeis pelo algoritmo de entrelaçamento Adam7 (formato PNG) com 7 passagens. Os tempos são contados em relação ao início da apresentação da imagem.

1.4 Full Color e True Color

O avanço tecnológico permitiu que, sucessivamente, fossem sendo comercializados equipamentos permitindo cada vez mais píxeis e cores. A designação *High Color* corresponde à possibilidade de utilizar 16 bits para cada cor ou intensidade de tons cinzentos, e permite o emprego de até 65536 cores diferentes numa imagem. Por seu lado, a designação *True Color* (cores reais) corresponde a equipamentos que empregam 24 bits para representar uma cor, ou seja, é possível apresentar imagens que empreguem

até 16.777.216 (2^{24}) cores. Este último número de cores é mais do que suficiente para representar imagens de qualidade fotográfica, dado que este número é muito superior à capacidade de descriminação de cores da visão humana[9]. No entanto, convém notar que existem domínios de aplicação que requerem um ainda maior número de cores, como acontece nos arquivos de pinturas. As imagens destes arquivos têm que permitir o emprego de um número de cores muito para além dos limites da visão humana. Para este fim existem actualmente formatos que empregam 48 bits por píxel (2^{48} cores).

As imagens dos tipos *High Color* e *True Color* não empregam mapas de cores, pois o mapa de cores de uma imagem com 24 bits por píxel poderia ocupar 50.331.648 bytes. Este espaço é muito maior do que o espaço ocupado por uma imagem de muito alta resolução de 4096×3072 píxeis e 24 bits por píxel (37.748.736 bytes).

1.5 Compressão

Como vimos anteriormente, o espaço ocupado por uma imagem com 24 (ou mesmo 16) bits por píxel pode ser demasiado grande, bastando para tal considerar que uma imagem de 800×600 píxeis e 24 bits por píxel ocupa 1.440.00 bytes, mesmo se considerarmos a capacidade actual das unidades de disco rígido. Este problema, que não é de hoje, verificava-se já durante a transmissão de imagens de dimensão e número de cores reduzidos, embora, nessa altura, a ênfase do problema residisse no tempo de transmissão.

A solução deste problema consiste em compactar a informação das imagens, comprimindo essa informação. Existem muitos algoritmos de compressão de informação, mas nem todos são adequados à compressão de imagens. Mesmo aqueles que o são produzem resultados variáveis porque a natureza das imagens pode variar muito. Consideremos duas imagens de igual dimensão e número de bits por píxel, uma contendo um gráfico de linhas e outra contendo uma imagem de qualidade fotográfica. O número de cores empregues por cada uma destas imagens é muito diferente. Enquanto um gráfico poucas vezes necessita de mais do que 8 cores, uma imagem de qualidade fotográfica emprega usualmente centenas de milhar de cores. Mas, a diferença fundamental reside em que a imagem contendo o gráfico apresenta grandes grupos de píxeis consecutivos em que todos os píxeis apresentam a mesma cor, enquanto a ocorrência de tais grupos é rara, senão inexistente, na imagem de qualidade fotográfica.

A ocorrência de grupos de píxeis consecutivos da mesma cor propicia o emprego de algoritmos de compressão do tipo RLE (Run Length Encoded) que substituem tais grupos por duas unidades de informação: o número de píxeis do grupo e o índice de cor dos seus píxeis.

O formato de imagem DIB emprega esta técnica a grupos de 8 ou 4 bits, consoante a imagem seja a 256 ou 16 cores, respectivamente. A implementação do algoritmo RLE pelo formato DIB define dois modos: codificado e absoluto.

Com grupos de 8 bits, e em modo modificado, cada unidade codificada é representada por dois bytes. O primeiro byte contém o número de píxeis consecutivos a serem representados pelo índice de cor contido no segundo byte. Neste modo, quando o valor do primeiro byte for 0, assume-se que a unidade codificada representa uma sequência

[9] Note-se que, face a este número de cores, a diferença entre uma imagem em formato digital e uma imagem fotográfica reside no número de píxeis da imagem digital, que é ainda inferior.

especial que é determinada pelo valor do segundo byte. Se este valor for 0, a sequência especial assinala que se atingiu o fim de uma linha da imagem. O fim da imagem (todas as linhas já foram processadas) é assinalado pelo valor 1 deste segundo byte. No caso em que o valor do segundo byte é 2, os dois bytes que se seguem à sequência indicam a posição do próximo píxel a processar relativamente ao píxel corrente (o último píxel processado). O primeiro destes dois bytes indica o número da linha do próximo píxel a processar (1 significa na linha seguinte), enquanto o segundo byte indica o número da coluna desse próximo píxel relativamente à coluna corrente (0 significa na mesma coluna).

Em modo absoluto, o primeiro byte encontrado tem o valor 0 e o segundo byte contém um valor compreendido entre 3 e 255, inclusive[10]. Este segundo byte representa o número de bytes que se seguem e em que cada um deles contém o índice da cor do respectivo píxel.

Quando o algoritmo RLE é aplicado a grupos de 4 bits, a única diferença para o algoritmo RLE aplicado a grupos de 8 bits consiste em que os bytes contendo índices de cor armazenam os índices de dois píxeis consecutivos no mesmo byte. Os 4 bits mais significativos contêm o índice do próximo píxel, enquanto os 4 bits menos significativos contêm o índice de cor do píxel que se lhe segue.

Existem outras famílias de algoritmos de compressão, como é o caso da família LZ[11], que detectam sequências de píxeis que, embora não apresentem todos os píxeis da mesma cor, apresentam a mesma sequência de cores. Todos estes algoritmos permitem reduzir o espaço ocupado por imagens contendo esquemas, diagramas ou linhas para, pelo menos, metade do espaço originalmente ocupado, chegando-se em alguns casos a obter factores de compressão de 10 ou superiores. É fácil verificar o impacte de uma compressão deste género no tempo necessário à transmissão da imagem de 360×270 píxeis com 256 cores que temos vindo a considerar[12].

Numa imagem digital de qualidade fotográfica, é raro encontrar as sequências de píxeis que tornam atraente o emprego dos algoritmos anteriores. Uma imagem de qualidade fotográfica caracteriza-se pela variação suave e contínua da cor (*continuous tone*) de píxel para píxel que é exactamente a situação mais desfavorável ao emprego de algoritmos de compressão das famílias RLE ou LZ. A variação contínua e suave é muito mais favorável ao emprego de outro tipo de algoritmos, como os baseados na Transformada Discreta do Co-seno (*Discrete Cosine Transform*), derivada da transformada de Fourier.

Matematicamente, a Transformada Discreta do Co-seno é uma transformada totalmente reversível, isto é, o resultado da transformada inversa é igual ao que foi transformado pela transformada directa. Na prática, existem perdas de carácter numérico que ocorrem durante a realização de operações aritméticas com representação limitada do número de algarismos dos operandos e perdas originadas pela eliminação de termos[13]. Quanto menos termos das séries forem considerados, maior será o erro cometido, embora o número de coeficientes da transformada e, consequentemente, o espaço para armazenar

[10] Isto permite diferenciar entre modo absoluto e modo codificado.
[11] O formato de imagem GIF emprega a variante LZW desta família, enquanto o formato PNG emprega por sua vez um algoritmo derivado do algoritmo LZ77.
[12] O tempo de transmissão seria de 23 segundos com um factor de compressão de 2 e de 4,6 segundos com um factor de 10.
[13] Na norma JPEG, a eliminação de termos é realizada pela quantização dos coeficientes da transformada.

os coeficientes, seja menor. A imagem será assim guardada em menor espaço, mas perderá informação e qualidade.

Por este motivo, este tipo de compressão é uma compressão com perda. A escolha da perda a admitir resulta de um compromisso entre o espaço a ocupar pela imagem comprimida e a maior ou menor qualidade que se pretenda das imagens comprimidas. Menor perda de qualidade significa maior espaço e vice-versa. Esta perda de qualidade é permitida pelo limite de capacidade da visão humana em descriminar cores. Com efeito, a visão humana tende a integrar as cores de áreas de reduzidas dimensões, interpretando essas áreas como se possuíssem uma única cor se tais áreas se apresentarem segundo um arco inferior a um dado limite. Por outro lado, a visão humana não distingue cores muito próximas umas das outras no espectro visível. Deste modo, duas imagens de qualidade fotográfica, uma com perda e outra sem perda, podem ser interpretadas como se da mesma imagem se tratasse, permitindo o emprego da compressão com perda.

Uma outra característica da visão humana permite reduzir ainda mais a informação contida numa imagem de qualidade fotográfica. Esta característica consiste no facto de a visão humana ser muito mais sensível à quantidade de luz (a luminância, em termos técnicos) do que à cor (a cromaticidade), o que significa que, se empregarmos um modelo de cor apropriado, como o modelo YCbCr, poderemos não empregar uma precisão igual para representar a luminância e a cromaticidade. Isto permite reduzir o espaço ocupado pela informação referente a cada píxel da imagem e, assim, o espaço ocupado por toda a imagem. A perda de informação que ocorre é aceitável porque não é detectada pela visão. Esta estratégia é adoptada pelo formato JFIF (JPEG).

2 Formatos PBM

A designação de formato de imagem PBM (Portable Bitmap) engloba três formatos de imagem para imagens a preto e branco, em escala de tons cinzentos e a cores, todos eles sem compressão e que apresentam uma estrutura comum[14]. Estes três tipos de formato de imagens são

- PBM (Portable BitMap) – imagens a preto e branco (sem tons de cinzento)
- PGM (Portable GrayMap) – imagens em tons de cinzento
- PPM (Portable PixMap) – imagens a cores

A definição original destes formatos teve em vista permitir a transmissão de imagens por meio de correio electrónico que, à data da definição, ainda não permitia a transmissão de ficheiros anexados, binários ou não. Os formatos PBM, PGM e PPM representavam então os conteúdos das respectivas imagens por meio de caracteres ASCII representáveis. Esta característica permitia a inserção de uma imagem numa mensagem de correio electrónico como se de texto se tratasse, mas tinha como consequência que o tamanho dos ficheiros fosse demasiado grande. A definição do formato foi mais tarde modificada para permitir a representação binária dos conteúdos das imagens.

[14] Por vezes, erradamente, estes três tipos de formato são designados por formato PPM.

2.1 Especificação dos formatos PBM

Os formatos de imagem PBM são constituídos pelos seguintes campos:

- Identificador do tipo de formato (designado por "magic number"), de acordo com

Tipo	ASCII	Binário
PBM	P1	P4
PGM	P2	P5
PPM	P3	P6

- Espaço em branco[15]
- Largura da imagem, em píxeis e em notação decimal, formatada em caracteres ASCII
- Espaço em branco
- Altura da imagem, em píxeis e em notação decimal, formatada em caracteres ASCII
- Espaço em branco

Se a imagem for dos tipos PGM ou PPM

- Valor máximo em notação decimal dos tons de cinzento (PGM) ou das componentes de cor (PPM), formatado em caracteres ASCII.
- Espaço em branco

Para todos os tipos de formato segue-se

- Valores dos píxeis da imagem, em número igual à altura da imagem vezes a sua largura para os tipos PBM e PGM, e três vezes este número para o tipo PPM, uma vez que cada píxel é representado pelas três componentes RGB da respectiva cor.

A ordenação dos valores dos píxeis corresponde ao varrimento das imagens linha a linha, de cima para baixo, e da esquerda para a direita em cada uma das linhas.

Quando o conteúdo da imagem for representado em ASCII, os valores correspondentes aos píxeis serão apresentados em notação decimal e separados por espaços em branco, marcas de tabulação ou marcas de fim de linha.

As variantes binárias dos tipos PBM, PGM e PPM armazenam os valores correspondentes aos píxeis das imagens em caracteres (bytes) contíguos, sem qualquer separador entre valores consecutivos. O tipo PBM combina os valores de grupos de 8 píxeis consecutivos num único carácter (byte). Os restantes tipos, PGM e PPM, fazem corresponder a cada píxel um carácter (PGM) ou três caracteres[16] (PPM).

15 Este espaço pode ser constituído por um qualquer número de caracteres em branco, TABs, CRs e LFs. Alguns produtos poderão não suportar caracteres TAB.

16 Um carácter (byte) por cada componente RGB do píxel, restringindo assim o valor máximo dos tons de cinzento e das componentes de cor a 255.

Para as variantes ASCII dos três tipos de formato, aplicam-se ainda as seguintes regras:

- É permitida a inserção de comentários em qualquer parte do ficheiro. O início de um comentário é assinalado por um carácter "#" e todo o texto desde este carácter até ao fim da respectiva linha é interpretado como texto do comentário.
- O comprimento máximo de cada linha está limitado a 70 caracteres[17].

2.2 Exemplos

Apresentamos de seguida exemplos de imagens formatadas nos formatos PBM. O primeiro exemplo é o de uma imagem a preto e branco de 24 píxeis de largura e 7 píxeis de altura em formato PBM, na sua variante ASCII, que figura 2.1 apresenta em ampliação. Note-se a inclusão de comentários.

```
P1
# feep.pbm
24  7
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1 0 0 1 1 1 1 0 0 1 1 1 1 0 0 1 1 1 1 0
0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 1 0
0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 0 0 0 1 1 1 1 0
0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0
0 1 0 0 0 0 0 1 1 1 1 0 0 1 1 1 1 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Figura 2.1 – Imagem em formato PBM (ampliada) do exemplo do texto.

Figura 2.2 – Imagem PGM (ampliada) do exemplo do texto.

[17] Esta limitação deriva da especificação inicial visando a transmissão de imagens por correio electrónico.

A variante binária da representação desta imagem é (o conteúdo da imagem apresenta os respectivos bytes codificados em notação hexadecimal)

```
P4
24      7
00000079E79E41041271C71E41041041E790000000
```

Como é óbvio, o tamanho da variante binária é muito mais compacto. O carácter de nova linha foi usado como separador entre os campos do identificador, altura, largura e conteúdo da imagem. Se em vez deste carácter tivéssemos empregue um espaço em branco, teríamos obtido

```
P4 24 7 00000079E79E41041271C71E41041041E790000000
```

Vejamos agora o exemplo de uma imagem em tons de cinzento, tipo PGM, que a figura 2.2 apresenta em ampliação, com a largura de 24 píxeis e a altura de 7 píxeis, nas suas variantes ASCII

```
P2
# feep.pgm
24      7
15
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  3  3  3  3  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15  15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0 15  0
0  3  3  3  0  0  0  7  7  7  0  0  0 11 11 11  0  0  0 15  15 15 15  0
0  3  0  0  0  0  0  7  0  0  0  0  0 11  0  0  0  0  0 15  0  0  0  0
0  3  0  0  0  0  0  7  7  7  7  0  0 11 11 11 11  0  0 15  0  0  0  0
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 binária (empregando notação hexadecimal para o conteúdo da imagem)

```
P2 24 7 15
000000000000000000000000333300777700BBBB00FFFF0030000070
0000B00000F00F00333000777000BBB000FFFF00300000700000B000
00F0000030000777700BBBB00F0000000000000000000000000000
```

Finalmente, apresenta-se o exemplo de uma imagem em formato PPM, que a figura 2.3 apresenta em ampliação,

```
P3
# feep.ppm
4 4
15
0  0  0  0  0  0  0  0  0 15  0 15
0  0  0  0 15  7  0  0  0  0  0  0
0  0  0  0  0  0  0 15  7  0  0  0
15  0 15  0  0  0  0  0  0  0  0  0
```

que, na variante binária, é

```
P3 4 4 15 000000000F0F0000F70000000000000F7000F0F000000000
```

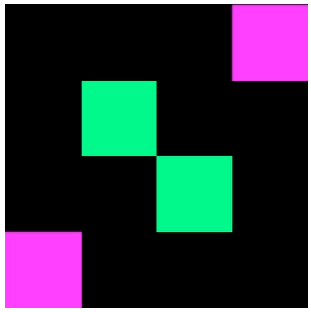


Figura 2.3 – Imagem (ampliada) em formato PPM do exemplo do texto.

3 Formato DIB

O formato DIB (Device Independent Bitmap), também designado por formato BMP, é um formato proprietário da Microsoft e é suportado por todas as variantes do sistema operativo Windows. Na origem da designação DIB está o facto de que este formato descreve as cores de uma forma que é independente do processo empregue por cada placa gráfica e respectivo monitor para apresentar a cor dos píxeis das imagens.

No formato DIB, as imagens são descritas por varrimento ascendente das linhas (de baixo para cima), sendo os píxeis de cada linha varridos da esquerda para a direita. Do varrimento de cada linha deve resultar uma sequência de bytes cujo número deve ser múltiplo de 4[18].

O formato DIB permite descrever imagens a cores com 1, 4, 8 ou 24 bits por píxel, representando assim imagens com 2, 16, 256 ou 2^{24} cores, respectivamente, empregando um mapa de cores em todos os casos excepto no último (2^{24} cores). O formato DIB permite ainda a compressão opcional do conteúdo de imagens com16 ou 256 cores pelo algoritmo RLE (Run Length Encoding) adaptado ao número de bits por píxel (4 ou 8).

A informação contida num ficheiro escrito em formato DIB está organizada em três blocos de informação, que só podem ser inseridos uma única vez, e têm que ser colocados no ficheiro pela seguinte ordem:

- Bitmap File Header, que descreve o ficheiro.
- Bitmap Info, que descreve o bitmap da imagem e é constituído pelos sub bloco Bitmap Info Header, que descreve a imagem contida no ficheiro, e pelo sub bloco Win3ColorTable que contém o mapa de cores da imagem se esta não empregar 24 bits por píxel.
- Dados da imagem, constituídos pelos píxeis da imagem segundo um varrimento de linhas de baixo para cima, e da esquerda para a direita em cada linha, comprimido ou não por meio do algoritmo RLE.

Todos os valores inteiros, sejam eles de 4 ou 2 bytes, encontram-se armazenados com o byte menos significativo em primeiro lugar (LSB first), conforme é uso do sistema operativo Windows.

[18] Para isto, os codificadores de imagens em formato DIB devem inserir um número de bytes sem significado tal que o número de bytes de cada linha seja múltiplo de 4.

3.1 Bloco Bitmap File Header

Este bloco descreve o ficheiro DIB corrente, ocupa 14 bytes, e tem a seguinte estrutura (em linguagem C)

```
typedef struct tagBITMAPFILEHEADER /*  bmfh  */
    {  UINT   bfType;
       DWORD  bfSize;
       UINT   bfReserved1;
       UINT   bfReserved2;
       DWORD  bfOffBits;
    } BITMAPFILEHEADER;
```

Os membros desta estrutura têm o seguinte significado:

bfType	Declara que o ficheiro corrente é um ficheiro em formato DIB. Contém obrigatoriamente o valor decimal 19778 (4D42 hexadecimal), que corresponde à cadeia de caracteres "BM" em ASCII.
bfSize	Comprimento total do ficheiro em bytes.
bfReserved1 bfReserved2	Estes campos estão reservados e, por isso, devem conter o valor 0.
bfOffBits	Número de bytes que, no ficheiro, precedem o primeiro byte correspondente ao conteúdo da imagem[19].

3.2 Bloco Bitmap Info

O bloco Bitmap Info (bloco descritor do bitmap) tem a estrutura

```
typedef struct tagBITMAPINFO /*  bmi  */
    {  BITMAPINFOHEADER  bmiHeader;
       RGBQUAD           bmiColors[];
    } BITMAPINFO;
```

que descreve a informação sobre o bitmap e, se presente, o mapa de cores.

Esta estrutura não é guardada no ficheiro contendo a imagem. O que é guardado nesse ficheiro é a informação constante da estrutura contendo a informação sobre a imagem (BITMAPINFOHEADER) e o mapa de cores da imagem, se esta não empregar 24 bits por píxel.

3.3 Sub Bloco Bitmap Info Header

O sub bloco Bitmap Info Header, como o comprimento de 40 bytes, é colocado no ficheiro imediatamente a seguir ao bloco Bitmap File Header. Um sub bloco Bitmap Info Header contém informação como o tamanho da imagem. A estrutura em linguagem C correspondente a este sub bloco é

```
typedef struct tagBITMAPINFOHEADER /*  bmih  */
    {  DWORD  biSize;
       LONG   biWidth;
       LONG   biHeight;
```

[19] Este valor é igual ao número de cores descritas pelo mapa de cores vezes 4, acrescido dos comprimentos do bloco descritor do ficheiro (14 bytes) e do bloco descritor da imagem (40 bytes).

```
    WORD   biPlanes;
    WORD   biBitCount;
    DWORD  biCompression;
    DWORD  biSizeImage;
    LONG   biXPelsPerMeter;
    LONG   biYPelsPerMeter;
    DWORD  biClrUsed;
    DWORD  biClrImportant;
} BITMAPINFOHEADER;
```

cujos membros assumem os seguintes significados

biSize	Espaço, em bytes, ocupado por esta estrutura de dados, ou seja, 40 bytes.
biWidth	Largura da imagem, em píxeis.
biHeight	Altura da imagem, em píxeis.
biPlanes	Define o número de planos que a imagem irá ocupar na unidade gráfica de saída. Este campo deve ter o valor 1, obrigatoriamente.
biBitCount	Número de bits por píxel que, obrigatoriamente, deve ser 1, 4, 8 ou 24.
biCompression	Define o tipo de compressão RLE realizada ao conteúdo da imagem e assume um dos valores BI_RGB (0, a imagem não está comprimida), BI_RLE8 (1, imagem comprimida em grupos de 8 bits) ou BI_RLE4 (2, imagem comprimida em grupos de 4 bits).
biSizeImage	Tamanho, em bytes, ocupado pelos dados da imagem. Este valor deve obrigatoriamente reflectir o comprimento dos dados da imagem depois de estes serem comprimidos, ou ser 0 se biCompression assumir o valor BI_RGB.
biXPelsPerMeter biYPelsPerMeter	Resolução horizontal e resolução vertical, em píxeis por metro, da imagem no dispositivo gráfico de saída[20]. Estes valores são ignorados se forem 0.
biClrUsed	Número de cores do mapa de cores referenciadas pelos dados da imagem. Se for 0, as aplicações devem assumir que a imagem referencia o número máximo de cores permitido pela profundidade dos respectivos píxeis (valor do membro biBitCount).
biClrImportant	Número de cores do mapa de cores que são consideradas importantes para a apresentação correcta da imagem. Considera-se que todas as cores são importantes quando este valor for 0.

[20] As aplicações que apresentam as imagens podem utilizar estes valores para melhor mapearem as dimensões reais das imagens aos dispositivos gráficos de saída.

3.4 Sub Bloco do Mapa de Cores

O mapa de cores destina-se a definir as componentes RGB de cada cor empregue pela imagem e a atribuir a cada cor definida no mapa o respectivo índice. Cada cor do mapa de cores é definida por meio da estrutura

```
typedef struct tagRGBQUAD /*  rgbq  */
    {  BYTE   rgbBlue;
       BYTE   rgbGreen;
       BYTE   rgbRed;
       BYTE   rgbReserved;
    } RGBQUAD;
```

Note-se a inversão da ordem normal das componentes RGB nesta estrutura.

A cada um dos membros rgbBlue, rgbGreen e rgbRed é atribuído um valor de intensidade compreendido entre 0 e 255. O membro rgbReserved não é empregue, pelo que o seu valor deverá ser 0.

Os índices de cor são atribuídos às cores segundo a ordem pela qual as cores ocorrem no mapa de cores. Assim, o índice 0 é atribuído à primeira cor constante do mapa, o índice 1 à segunda cor e, assim, sucessivamente.

Não existe mapa de cores sempre que um ficheiro em formato DIB contenha uma imagem cujos píxeis ocupem 24 bits. Neste caso, os três bytes representando um píxel contêm as intensidades das componentes RGB da sua cor.

3.5 Bloco Bitmap DIB

Como atrás foi dito, o conteúdo do bitmap de uma imagem em que a informação de cada píxel está contida em 24 bits é constituído pelas componentes da cor de cada píxel, sendo os píxeis da imagem ordenados segundo linhas varridas da esquerda para a direita e de baixo para cima, não sendo permitida qualquer compressão.

Para os bitmaps com 4 bits por píxel é permitida a compressão pelo algoritmo RLE aplicado a grupos de 4 bits. Os bitmaps de 8 bits por píxel (256 cores) podem também ser comprimidos pelo algoritmo RLE com base em grupos de 8 bits.

4 Formato GIF

O formato Graphics Interchange Format (GIF) foi inicialmente concebido para a transmissão de imagens através das linhas de comunicação de muito baixo débito existentes na altura[21]. O formato GIF é propriedade da CompuServe Inc.

O formato permite armazenar ou transmitir imagens com um máximo de 256 cores, definidas sempre por meio de mapas de cor. Cada píxel de uma imagem no formato GIF contém o índice correspondente ao número de ordem da sua cor no mapa de cores. O conjunto dos índices que compõem cada imagem está comprimido pelo algoritmo de Lempel-Ziv Welch (LZW), que é um algoritmo de compressão sem perda.

Na base do formato GIF está o conceito de Data Stream, ou canal de dados. O formato é na realidade um protocolo entre uma fonte emissora de imagens e uma aplicação de

[21] Modems de 2 ou 4 kbit/s e linhas telefónicas de comutação analógica.

destino que realiza a apresentação das imagens. Quando a fonte emissora se encontra armazenada num ficheiro, dizemos que estamos perante um ficheiro em formato GIF.

Um ficheiro, ou canal de dados, no formato GIF pode conter mais do que uma imagem. Blocos de controlo inseridos entre imagens sucessivas determinam o tempo durante o qual cada imagem persistirá na unidade gráfica de saída, permitindo assim a apresentação sequenciada de várias imagens. Se as imagens constituírem uma sequência animada, estaremos perante o que se designa usualmente por GIFs animados (*animated GIFs*).

Os índices de cor de uma imagem no formato GIF estão ordenados segundo uma sequência que resulta do varrimento das linhas da imagem da esquerda para a direita em cada linha e de cima para baixo no conjunto de linhas da imagem. Este varrimento vertical pode ser por linhas consecutivas ou entrelaçado.

4.1 Estrutura de Um Canal de Dados GIF

A informação transmitida por um canal de dados GIF está organizada em blocos, denominados Blocos Lógicos, que pertencem a três tipos:

- Blocos de Controlo (Control Blocks)
- Blocos de Processamento de Imagem (Image Rendering)
- Blocos Especiais (Special Purpose)

A descrição dos blocos do formato GIF que a seguir apresentamos não é, nem pretende ser, exaustiva e contém apenas os blocos de maior utilização ou aqueles cuja presença num canal de dados GIF é obrigatória.

4.2 Blocos de Controlo

Os blocos de controlo contêm a informação necessária para que a aplicação que processa a informação transmitida por num canal de dados organizado segundo o formato GIF possa processar esses dados e controlar as unidades periféricas de representação gráfica, dividindo-se em:

- Header (Cabeçalho)
- Logical Screen Descriptor
- Graphic Control Extension
- Trailer

Os blocos de controlo não contêm qualquer informação sobre o conteúdo, dimensões ou posicionamento das imagens.

Header

Este bloco de controlo identifica o início de um fluxo de dados em formato GIF. A sua presença é obrigatória em todos os canais ou ficheiros, só podendo existir um bloco Header por cada canal ou ficheiro. A sua estrutura é

Campo	Bytes	Conteúdo
Assinatura	3	Cadeia de caracteres “GIF”, em ASCII.
Versão	3	Determina a versão do formato GIF empregue e é constituída por uma cadeia de 3 caracteres que pode ser “87a” ou “89a”, em ASCII.

Trailer

Este bloco de controlo assinala o fim de um fluxo de dados em formato GIF e é constituído por um único byte contendo o valor hexadecimal 3B.

Logical Screen Descriptor

Em cada canal de dados ou ficheiro segundo o formato GIF, tem que existir um bloco do tipo Logical Screen Descriptor, obrigatoriamente colocado logo a seguir ao bloco Header. Um bloco deste tipo tem por missão definir as características e configuração que a unidade gráfica de saída deve apresentar ou possuir e os valores de parâmetros globais para todo o processamento.

A estrutura de um bloco Logical Screen Descriptor contém os seguintes campos

Campo	Bytes	Conteúdo
Logical Screen Width	2	Valor sem sinal da largura, em píxeis, da área da unidade gráfica de saída onde as imagens serão apresentadas.
Logical Screen Height	2	Valor sem sinal da altura, em píxeis, da área da unidade gráfica de saída onde as imagens serão apresentadas.
Packed Fields	1	Conjunto de 8 bits com a forma GCCCSTTT (ver significado mais à frente).
Background Color Index	1	Índice da cor de fundo no mapa global de cores, se o mapa existir. Esta cor é aplicada à área de apresentação das imagens.
Pixel Aspect Ratio	1	Quociente entre as dimensões físicas da altura e largura dos píxeis na unidade gráfica onde as imagens foram geradas. Este valor, variável de unidade para unidade gráfica, destina-se a evitar a deformação das imagens. Se for 0, a aplicação deve assumir que a largura e a altura dos píxeis são iguais.

O campo Pixel Aspect Ratio permite evitar a deformação das imagens quando o quociente entre as dimensões físicas da altura e da largura dos píxeis variar de unidade para unidade gráfica. O valor contido neste campo é calculado pela expressão

$$Aspect\ Ratio = \frac{Pixel\ Aspect\ Ratio + 15}{64} \tag{4.1}$$

A informação contida no campo Packed Fields deve ser interpretada da seguinte forma:

Bits	Significado
G	Assinala se, logo a seguir ao bloco Logical Screen Descriptor, existe (1) ou não (0) um mapa global de cores a aplicar às imagens que não possuam mapa local.
CCC	O valor contido nestes bits mais 1 corresponde ao número de bits que cada componente de cor ocupa. Exemplo: se cada componente de cor poder assumir 16 intensidades (ocupar 4 bits), o valor de CCC será 011 (binário).
S	Se existir um mapa global de cor, o valor deste bit assinala se as cores desse mapa estão ordenadas por ordem decrescente de frequência de emprego (1) ou não (0).
TTT	Se existir um mapa global de cores, o valor destes bits mais 1 indica o número de cores definidas por esse mapa, igual a $2^{(TTT+1)}$. Estes bits devem ser sempre preenchidos mesmo que não exista mapa global de cores para que a aplicação possa configurar a unidade gráfica de saída da forma mais conveniente para a apresentação das imagens que se seguem. Exemplo: se o valor contido em TTT for 011 (binário), o número de cores definidas no mapa global de cores é 2^4, ou seja, 16 cores.

Graphics Control Extension

Um bloco do tipo Graphics Control Extension tem por objectivo controlar a apresentação da próxima imagem contida no canal de dados ou ficheiro em formato GIF. A presença de um bloco deste tipo é opcional.

A estrutura de informação de um bloco Graphics Control Extension é

Campo	Bytes	Conteúdo
Extension Introducer	1	Contém o valor hexadecimal 21.
Graphic Control Label	1	Contém o valor hexadecimal F9.
Block Size	1	Número de bytes deste bloco que se seguem a este campo, sem incluir o comprimento do campo terminador. Para um Graphics Control Block, este campo assume o valor 4.
Packed Fields	1	Bits organizados como RRRDDDUT.
Delay Time	2	Valor sem sinal do tempo (em centésimas de segundo), a contar do fim da apresentação completa da próxima imagem, durante o qual o processamento de dados deve ser suspenso antes de prosseguir. Não existe espera quando este valor for 0.

Transparent Color Index	1	Índice da cor que deve ser interpretada como cor transparente. Todos os píxeis a que corresponda este índice de cor não serão processados. O valor contido neste campo só terá significado se o bit T do campo Packed Fields estiver aceso (1).
Block Terminator	1	Este byte, com valor 0, assinala o fim do bloco.

O significado dos bits do campo Packed Fields (RRRDDDUT) é o seguinte

Bits	**Significado**
RRR	Bits reservados e, por isso, devem estar apagados (0).
DDD	Disposal Method, cujo valor indica o que a aplicação deve fazer quando terminar a apresentação da próxima imagem, que pode ser: 0 – o processamento prossegue imediatamente 1 – a imagem não é apagada, o processamento prossegue imediatamente 2 – a imagem é apagada, atribuindo a cor de fundo a todos os seus píxeis, e o processamento prossegue logo de seguida 3 – a imagem é apagada, substituindo o conteúdo da sua área pelo conteúdo que lá se encontrava antes da apresentação da imagem 4 a 7 – valores por definir
U	Indica se o processamento deve esperar por qualquer acção do utilizador antes de prosseguir (1) ou não (0). Quando este bit for 1 e o valor do campo Delay Time for diferente de 0, o processamento prosseguirá logo que o utilizador execute qualquer acção ou o tempo de espera termine.
T	Indica se o campo Transparency Index contém (1) ou não (0) o valor do índice da cor a processar como cor transparente.

4.3 Blocos de Processamento de Imagem

Os blocos de Processamento de Imagem (*Image Rendering*) destinam-se a declarar os parâmetros próprios de cada imagem e os respectivos conteúdos, dividindo-se em:

- Image Descriptor
- Color Table
- Table Based Image Data
- Plain Text Extension

Veremos apenas os três primeiros tipos de bloco, já que o último tipo não é praticamente usado[22].

[22] O bloco Plain Text Extension descreve o conteúdo da imagem empregando apenas caracteres ASCII representáveis, o que aumenta o comprimento deste tipo de bloco em relação ao tipo

Cada imagem contida num fluxo de dados segundo o formato GIF é descrita por um bloco Image Descriptor, seguido opcionalmente por um bloco Color Table contendo o mapa local de cores para essa imagem (e só ela) e, finalmente, por um bloco do tipo Table Based Image Data ou Plain Text Extension com o conteúdo da imagem.

A definição de um mapa local de cor para cada imagem é opcional. Se este mapa local existir, a imagem que se lhe segue será apresentada de acordo com as cores definidas pelo mapa. Quando não existir mapa local de cores, a imagem será apresentada com as cores definidas pelo mapa global de cores que, neste caso, será de presença obrigatória.

Tal como já vimos, os blocos definindo uma imagem e os seus parâmetros podem ser precedidos por um bloco Graphics Control Extension que controlará a forma como a imagem será eliminada (ou não) da unidade gráfica de saída e o tempo durante o qual a imagem perdurará nessa unidade.

Image Descriptor

Um bloco Descritor da Imagem (Image Descriptor) contém os parâmetros que caracterizam a imagem e o mapa local de cores, se este existir, que se lhe seguem. Os parâmetros da imagem deverão ser tais que a imagem caiba integralmente dentro do espaço de apresentação definido pelo bloco Logical Screen Descriptor do corrente fluxo de dados em formato GIF.

Os campos contidos num bloco Image Descriptor são:

Campo	**Bytes**	**Conteúdo**
Image Separator	1	Identifica o início de uma nova imagem. Contém o valor hexadecimal 2C.
Image Left Position	2	Valor sem sinal correspondente à distância horizontal, em píxeis, do canto superior esquerdo da imagem ao canto superior esquerdo da área de representação de imagens definida pelo bloco Logical Screen Descriptor.
Image Top Position	2	Valor sem sinal correspondente à distância vertical, em píxeis, do canto superior esquerdo da imagem ao canto superior esquerdo da área de representação de imagens definida pelo bloco Logical Screen Descriptor.
Image Width	2	Valor sem sinal da largura da imagem em píxeis.
Image Height	2	Valor sem sinal da altura da imagem em píxeis.
Packed Fields	1	Byte contendo os bits LISRRCCC.

Table Based Image Data, que é binário. Por esta razão, não é muito comum o seu emprego, tendo caído praticamente em desuso.

Os bits do campo Packed Fields assumem o seguinte significado:

Bits	Significado
L	Local Color Table Flag que assinala se um mapa local de cores precede a imagem (1) ou não (0).
I	Interlace Flag, que assinala se as linhas da imagem se encontram entrelaçadas (1) ou não (0).
S	Sort Flag, que indica se as cores definidas no mapa local de cores se encontram ordenadas por ordem decrescente de frequência do seu emprego pela imagem (1) ou não (0).
RRR	Reserved. Estes bits devem estar apagados (0).
CCC	Size of Color Table que representa o número de cores constantes do mapa de cor (mapa local, neste caso). O número de cores definidas pelo mapa é igual a $2^{(CCC+1)}$. Exemplo: se CCC for 100 (binário), o número de cores contidas no mapa será de 32 cores.

Color Table

Este tipo de bloco define um mapa de cores que pode ser global[23] ou local. Neste último caso, o bloco Color Table deve ser inserido logo a seguir ao bloco Image Descriptor e imediatamente antes do bloco onde se encontra o conteúdo da imagem.

Num mapa de cores, cada cor ocupa 3 bytes, um por cada componente RGB de cor. O índice de cada cor é atribuído pela ordem pela qual a cor figura no mapa. Assim, à cor definida pelos primeiros três bytes do mapa é atribuído o índice 0, à cor dos segundos três bytes é atribuído o índice 1 e, assim, sucessivamente.

Um mapa de cores só pode definir 2, 4, 8, 16, 64, 128 ou 256 cores. É erro definir um qualquer outro número de cores, pois entrar-se-ia em contradição com o significado dos bits correspondentes a Size of Color Table. O número de bytes ocupados por um mapa de cores é igual a três vezes o número de cores definidas pelo mapa.

Table Based Image Data

Cada bloco do tipo Table Based Image Data transporta o conteúdo total de uma imagem, onde cada píxel é representado pelo índice da respectiva cor. A cadeia de índices que assim se forma é obtida pelo varrimento das linhas da imagem da esquerda para a direita numa mesma linha e de cima para baixo no conjunto das linhas. Se se empregar entrelaçamento, as linhas são reordenadas de acordo com o respectivo algoritmo antes de se constituir a cadeia de índices. Uma vez formada a cadeia de índices, é-lhe aplicado o algoritmo LZW de compressão sem perda.

No fluxo de dados, a cadeia já comprimida é precedida por um byte cujo valor corresponde ao comprimento em bits das unidades que foram comprimidas. O algoritmo codificador que, basicamente, procura cadeias repetidas e lhes atribui códigos, dispõe de um número limitado de códigos e, por isso, pode chegar a esgotar todos os códigos de que dispõe. Nessa altura, o codificador deve inserir na cadeia comprimida um código

[23] Neste caso, o bloco Color Table deve ser obrigatoriamente colocado logo a seguir ao bloco Logical Screen Descriptor.

especial que assinala a reinicialização da tabela de códigos e, consequentemente, o algoritmo descodificador deve executar uma acção idêntica. Deve notar-se que o codificador é livre de assinalar uma reinicialização da tabela de códigos em qualquer altura, mesmo que ainda não tenha esgotado todos os códigos de que dispõe, devendo as implementações do algoritmo descodificador estar preparadas para esta eventualidade.

5 Formato PNG

O formato PNG (Portable Network Graphics) surgiu como resposta às limitações técnicas e às restrições legais derivadas dos direitos de propriedade do formato GIF. Com efeito, o limite de 256 cores deste formato deixou de responder às exigências dos utilizadores quando estes passaram a dispor de hardware gráfico permitindo mais do que 256 cores. Por outro lado, a controvérsia em torno dos direitos de propriedade do formato GIF e do algoritmo LZW de compressão que emprega, que obrigavam à obtenção de licenças para o desenvolvimento de aplicações empregando o formato GIF, limitavam muito os criadores de aplicações.

A definição do novo formato PNG reteve algumas das características mais vantajosas do formato GIF como:

- Suporte de imagens até 256 cores (imagens com 1, 2, 4, e 8 bits por píxel), empregando mapas de cor.
- Conceito de canal de dados gráficos, que permite a apresentação e transmissão sequenciada e controlada de imagens.
- Apresentação progressiva de imagens, que permite a sua apresentação antes de completada a sua transmissão, embora com baixo nível de detalhe[24].
- Transparência parcial, permitindo que partes da imagem sejam declaradas como transparentes[25].
- Informação textual, que permite a inclusão de textos e de comentários e o controlo da apresentação temporizada de imagens.
- Independência da plataforma de hardware e software.
- Compressão sem perda.

Em complemento, a definição do formato PNG acrescentou a seguinte funcionalidade:

- Imagens a cores reais (*True Color*), empregando até 48 bits por píxel, permitindo 24 ou 48 bits por píxel, ou seja, 8 ou 16 bits por cada componente da cor.
- Transparência por meio de um canal alfa, definindo máscaras de transparência globais, variando de píxel para píxel, nas imagens com 8 ou 16 bits por píxel, implicando o emprego de 1 ou 2 bytes adicionais por píxel, respectivamente, ou por componente de cor.

[24] A apresentação progressiva do formato PNG é um refinamento da técnica de entrelaçamento do formato GIF, mas aplicada a píxeis e não a linhas.

[25] O formato GIF só permite declarar uma única cor como cor transparente, não existindo controlo do espaço de imagem que é tornado transparente.

- Informação (opcional) sobre a correcção gama aplicada à imagem, que, quando presente, permite que as aplicações de apresentação possam realizar a correcção adequada às cores das imagens, em função das características de gama das unidades de saída gráfica.
- Detecção da corrupção de dados, realizada em todos os blocos das imagens.
- Maior rapidez na apresentação de imagens através do algoritmo de entrelaçamento Adam7 para apresentação progressiva.
- Algoritmo de compressão do domínio público que, portanto, não se encontram sujeitos a restrições ou licenças derivadas de direitos de propriedade e que, na sua generalidade, permitem compactar a informação 5 a 25% mais eficientemente do que o algoritmo LZW do formato GIF[26].
- Ordenação única dos bytes segundo a ordem de transmissão por rede (network byte order), em que os bytes mais significativos precedem os bytes menos significativos (MSB first) e, portanto, torna o formato PNG independente da plataforma.

5.1 Organização de um canal (ou ficheiro) PNG

A informação contida num ficheiro PNG, ou transmitida por um canal de dados segundo o formato PNG, encontra-se estruturada em blocos, ou *chunks*, na terminologia deste formato. Estes blocos, em número variável, são precedidos pela assinatura do formato que, obrigatoriamente, é colocada no início do ficheiro ou é a primeira informação transmitida pelo canal de dados. Esta assinatura assinala univocamente que a informação que se lhe segue se encontra no formato PNG. É constituída por 8 bytes cujo valor é

Valor hexadecimal	Cadeia de caracteres (notação C)
89504E470D0A1A0A	\211PNG\r\n\032\n

A esta assinatura segue-se um número variável de blocos, dos quais o primeiro bloco é obrigatoriamente do tipo IHDR e o bloco final do tipo IEND. Entre estes dois blocos poderão ser inseridos um ou mais blocos do tipo IDAT com o conteúdo de imagens, e outros blocos como os blocos definidores de paletas (blocos PLTE).

Cada bloco é composto pelas 4 secções que a tabela 5.1 apresenta.

Apresentam-se se seguida os tipos de bloco mais comuns.

Bloco IHDR

Logo imediatamente a seguir à assinatura do formato PNG tem que ser inserido um bloco do tipo IHDR, único em todo o ficheiro ou canal de dados. O bloco IHDR, cujo identificador tem o valor hexadecimal 49484452 (IHDR, em caracteres ASCII), tem o comprimento fixo de 13 bytes.

[26] Em pequenas imagens, esta vantagem pode mesmo atingir 40 a 50%.

Secção	Bytes	Conteúdo
Comprimento da secção de dados do bloco	4	Inteiro sem sinal, com o número de bytes (n) contidos na secção de dados do bloco. Este valor pode ser 0.
Tipo do bloco	4	Valor inteiro que identifica o tipo de bloco e que, quando interpretado em ASCII, é constituído por 4 caracteres correspondentes a letras maiúsculas e minúsculas.
Dados do bloco	n	Conteúdo dos dados do bloco. Esta secção pode ser omitida, dependendo do tipo de bloco.
CRC	4	CRC de 4 bytes, calculado sobre todos os bytes do bloco que precedem esta secção, excluindo a secção contendo o comprimento do bloco.

Tabela 5.1 – Secções constituintes de um bloco (chunk) do formato PNG.

Color type	Bit depth	Conteúdo
0	1, 2, 4, 8 ou 16	Cada píxel da imagem (em tons de cinzento) contém a respectiva intensidade.
2	8 ou 16	Cada píxel da imagem, que é a cores, contém os valores das componentes da sua cor.
3	1, 2, 4 ou 8	Cada píxel da imagem contém um índice de cor. Um bloco PLTE deverá, obrigatoriamente, preceder quaisquer blocos IDAT.
4	8 ou 16	A cada píxel da imagem (em tons de cinzento) corresponde o valor da respectiva intensidade e o valor local da máscara de transparência.
6	8 ou 16	A cada píxel da imagem (a cores) estão associados os valores das suas três componentes e o valor local da máscara de transparência.

Tabela 5.2 – Valores permitidos nos campos Color Type e Bit Depth de um bloco IHDR do formato PNG.

Os campos definidos por um bloco IHDR dentro da sua secção de dados são

Campo	Bytes	Conteúdo
Width	4	Largura da imagem, em píxeis.
Height	4	Altura da imagem, em píxeis.
Bit depth	1	Número de bits por componente de cor ou por índice de cor. Pode assumir os valores 1, 2, 4, 8 ou 16.

Color type	1	Identifica o tipo de imagem, somando os valores 1, 2 e 4, de acordo com: 1 - a imagem emprega mapa de cores 2 - a imagem é colorida 4 - a imagem emprega o canal alfa para obter efeitos de transparência A tabela 5.2 apresenta os valores permitidos neste campo, conjuntamente com os valores do campo Bit Depth que lhes estão associados.
Compression method	1	Identifica o tipo de compressão aplicada aos dados do conteúdo das imagens (ver o bloco IDAT). Este campo tem obrigatoriamente o valor 0.
Filter method	1	Identifica que tipo de pré-processamento (filtro) foi aplicado às imagens antes da compressão do seu conteúdo. Este campo tem, obrigatoriamente, o valor 0.
Interlace method	1	Indica se os dados das imagens se encontram entrelaçados[27] segundo o método Adam7 (1) ou se não (0).

Bloco IEND

Este tipo de bloco assinala o fim do ficheiro ou canal de dados PNG. Tem um comprimento de dados nulo, pois não contém secção de dados. O valor hexadecimal do seu identificar é 49454E44, a que corresponde a cadeia de caracteres IEND em ASCII.

Bloco PLTE

Um bloco do tipo PLTE é um bloco que só pode aparecer uma única vez no ficheiro ou no canal de dados PNG, precedendo todos e quaisquer blocos IDAT. Contém um mapa de 1 a 256 cores a ser empregue por todas as imagens que se seguirem. Cada cor ocupa três bytes com os valores das suas componentes RGB. Ao primeiro conjunto de três bytes é atribuído o índice de cor 0, ao segundo conjunto o índice 1 e, assim, sucessivamente. O comprimento em bytes de um bloco PLTE é igual a três vezes o número de cores definidas pelo mapa de cores. Um comprimento que não seja múltiplo de 3 constitui um erro.

O valor hexadecimal do identificador de um bloco PLTE é 504C5445 (PLTE, em ASCII).

Este tipo de bloco deve ser inserido no canal de dados ou ficheiro PNG sempre que o valor do campo Color Type do bloco IHDR é 3 ou, opcionalmente, quando o valor de Color Type é 2 ou 6, com o fim de sugerir até 256 cores de substituição para o caso de a unidade gráfica de saída não suportar cores verdadeiras. Não é permitida a inserção de qualquer bloco PLTE quando o valor de Color Type for 0 ou 4.

O número de cores definidas por um bloco PLTE não pode exceder aquele que deriva do valor de Bit Depth declarado no bloco IHDR. Se, por exemplo, Bit Depth for 4, o número máximo de cores não poderá exceder 2^4, ou seja, 16 cores. No entanto, é

[27] O entrelaçamento segundo este método será apresentado mais adiante.

perfeitamente possível definir menos cores do que este limite, mas verificar-se –à um erro se uma imagem referenciar um índice que não conste do mapa de cores.

Bloco IDAT

Os blocos IDAT armazenam o conteúdo das imagens. Cada imagem pode ser descrita por mais do que um bloco IDAT mas, neste caso, os blocos IDAT da imagem devem aparecer consecutivamente no ficheiro ou canal de dados PNG, sem nenhum outro bloco que não seja do tipo IDAT colocado entre eles.

O identificador de um bloco IDAT tem o valor hexadecimal 48444154 (IDAT, em ASCII).

Quando não é empregue entrelaçamento para apresentação progressiva das imagens, as linhas de uma imagem são varridas de cima para baixo e, em cada linha, da esquerda para a direita. Antes da compressão é aplicado a cada linha um algoritmo de filtragem. Quando se emprega entrelaçamento, a fase de filtragem é precedida pelo rearranjo dos píxeis da imagem, de acordo com o algoritmo Adam7.

Bloco gIFg

O bloco gIFg é um tipo de bloco de extensão à definição do formato PNG e, consequentemente, a sua implementação e interpretação são facultativas. O interesse deste tipo de bloco reside no facto de transpor para o formato PNG a funcionalidade do bloco Graphic Control Extension do formato GIF89a, que permite obter animações simples controlando a apresentação de imagens sucessivas.

O valor hexadecimal do identificador de um bloco gIFg é 67494667 (gIFg, em ASCII) e contém os campos Disposal Method (1 byte), User Input Flag (1 byte) e Delay Time (2 bytes). O significado destes campos é, respectivamente, idêntico ao significado dos bits DDD e U do campo Packed Fields e ao campo Delay Time do bloco Graphics Control Extension do formato GIF.

5.2 Bloco gAMA

Este bloco, cujo valor hexadecimal do seu identificador é 67414D41 (gAMA, em ASCII), contém a informação referente às sucessivas correcções gama a que a imagem já foi sujeita, por forma a poder apresentar a imagem com cores correctas, independentemente da distorção de cor provocada pelo valor de gama da unidade gráfica de saída.

O conteúdo deste tipo de bloco é constituído por um único campo, com 4 bytes de comprimento, contendo o produto de 100.000 pelo valor acumulado das correcções gama a que a imagem foi sujeita. Assim, por exemplo, a um valor de correcção gama cujo expoente é 1/2,2, corresponde um valor deste campo igual a 1/2,2×100.000, ou seja, 0,45455×100.000, cujo valor inteiro é 45.454 (decimal).

5.3 Apresentação Progressiva

Designado por apresentação progressiva, o entrelaçamento suportado pelo formato PNG permite dar ao utilizador uma ideia da imagem total a partir de dados incompletos, através do algoritmo Adam7.

Verificação da Integridade dos Dados

O formato PNG especifica a verificação da integridade dos dados constantes de cada bloco por meio de um algoritmo CRC normalizado com pré e pós condicionamento, definido pela norma ISO 3309, que emprega o polinómio

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \qquad (5.1)$$

O registo CRC de 32 bits é inicializado a 1s e cada byte do bloco é processado do bit menos significativo (1) ao bit mais significativo (128). Depois de todos os bytes terem sido processados, o registo CRC é invertido e o seu valor transmitido no fim do bloco.

5.4 Algoritmo de Compressão e Descompressão

O algoritmo de compressão e descompressão empregue pelo formato PNG é o algoritmo utilizado pelos programas zip, gzip e pkzip, que deriva do algoritmo LZ77 e de que existem implementações do domínio público. As cadeias de caracteres comprimidos por este algoritmo seguem o formato zlib, cuja estrutura é

Campo	No de bytes
Compression methods/flags code	1
Additional flags/check bits	1
Compressed data blocks	(variável)
Check value	4

6 Formato JFIF (JPEG)

6.1 JPEG e JFIF

A norma internacional JPEG (ISO 10918-1) define uma família de algoritmos de compressão e descompressão, com e sem perda, para imagens de qualidade fotográfica, também designadas por imagens de tons contínuos (*continuous tone images*). Em troca de um menor comprimento das imagens, estes algoritmos aceitam a perda controlada de informação do conteúdo das imagens com base no facto de a visão humana ser incapaz de distinguir as muito pequenas diferenças de cor que existem entre píxeis contíguos nas imagens de qualidade fotográfica[28]. A sigla JPEG deriva do nome da comissão internacional de normalização, o Joint Photographic Experts Group, oficialmente designado por Joint Committee ISO/IEC JTC1 SC 29 Working Group 1.

Contrariamente ao que é do conhecimento comum, não existe nenhum formato JPEG porque a norma ISO 10918-1 não define tal formato. Esta norma limita-se a definir os algoritmos de compressão e descompressão, deixando aos formatos existentes a liberdade de aplicarem ou não os algoritmos normalizados. Um dos formatos que adoptou esta norma foi o formato TIFF (Tag-based Image File Format), propriedade da Aldus Corporation. A partir da versão 6 deste formato, os codificadores de imagens em formato TIFF passaram a poder empregar os algoritmos JPEG em paralelo com os

[28] Relembre-se que a visão humana integra as cores de píxeis contíguos quando o ângulo de visão de um grupo de píxeis é inferior a um valor limite.

outros algoritmos de compressão que o formato já empregava. No entanto, o emprego deste formato não se generalizou devido à sua complexidade que deriva dos inúmeros parâmetros de imagem que permite e que tornam complicada a sua implementação.

O vazio criado pela norma ISO 10918-1 foi preenchido pelo formato JFIF (JPEG File Interchange Format) que é um formato de imagem simples e que, embora apresente limitações, é relativamente fácil de implementar e permite imagens a cores reais (2^{24} cores)[29]. A aceitação deste formato foi enorme. Um dos factores que mais contribuíram para esta aceitação foi a expansão da World Wide Web cujos utilizadores necessitavam poder transmitir imagens com mais cores do que o máximo de 256 cores permitido pelo formato GIF, mas de comprimento reduzido para diminuir o respectivo tempo de carregamento.

A popularidade deste formato, associada à necessidade de abreviar o comprimento do seu nome (JPEG File Interchange Format), levaram o comum dos utilizadores a designá-lo simplesmente por JPEG e assim se estabeleceu a confusão entre o nome do formato JFIF e nome da norma JPEG.

6.2 Características do Formato JFIF

Contrariamente à maioria dos muitos formatos de imagem existentes, o formato JFIF não utiliza o modelo de cor RGB, mas o modelo YCbCr (luminância, crominância azul e crominância vermelha) com quantização da cor que, logo à partida, reduz o conteúdo da imagem a metade do seu tamanho original, mas acarreta perda de informação.

A compressão JPEG empregue por este formato utiliza algoritmos de compressão e descompressão do tipo da transformada discreta do co-seno, derivada da transformada de Fourier. Os algoritmos são aplicados a grupos de 8×8 píxeis de uma mesma componente de cor, sendo cada componente de cor processada independentemente das outras. Os coeficientes da transformada, designados por coeficientes AC e DC, depois de alguns tratamentos são armazenados em segmentos de informação dos ficheiros JFIF[30].

6.3 Formato dos Ficheiros JFIF

Os ficheiros em formato JFIF estão organizados por secções ou blocos separados entre si por marcas com o comprimento de um byte. Cada marca é precedida por um byte contendo o valor hexadecimal FF que assinala que se segue uma marca. A tabela 6.1 apresenta as marcas mais comuns presentes em ficheiros JFIF.

Todos os ficheiros contêm um segmento SOI no seu início, segmento este que é obrigatoriamente seguido por um segmento do tipo APP0. Um número variável de segmentos do tipo APPn poderá ser incluído logo a seguir ao segmento APP0. O ficheiro inclui seguidamente um ou mais segmentos DQT contendo cada um deles uma tabela de quantização, um segmento SOF0 descrevendo os parâmetros principais da imagem e um ou mais segmentos DHT contendo as tabelas de Huffman empregues na compressão e que são necessárias para a descompressão. A seguir a estes segmentos vem um segmento do tipo SOS que descreve as componentes comprimidas da imagem. O ficheiro termina com um segmento EOI que assinala o seu fim.

[29] O formato JFIF foi criado por Eric Hamilton da C-Cube Microsystems.
[30] Ver a descrição destes segmentos mais adiante.

Todas as marcas anteriormente citadas, à excepção das marcas SOI e EOI, são imediatamente seguidas por dois bytes cujo valor corresponde ao comprimento do respectivo segmento. Neste comprimento são contados os dois bytes do próprio comprimento, mas não os bytes introdutores e identificadores do segmento.

Segmento SOI

Este segmento destina-se a assinalar o início de um ficheiro no formato JFIF e, consequentemente, não tem informação sobre o seu comprimento, nem qualquer conteúdo. É constituído por apenas dois bytes cujo valor hexadecimal é FFD8.

Segmento EOI

Um segmento EOI assinala o fim de um ficheiro em formato JFIF e, tal como para o segmento SOI, contém apenas 2 bytes cujo valor hexadecimal é FFD9.

Marca	**Valor Hexadecimal**	**Tipo de marca**
SOI	D8	Início de uma imagem
APP0	E0	Segmento identificador JFIF
APPn	En	Outros segmentos identificadores opcionais
DQT	DB	Identificador de tabela de quantização
SOF0	C0	Início de um quadro (frame) da imagem
DHT	C4	Identificador de uma tabela de Huffman
SOS	DA	Início dos dados (conteúdo) de uma imagem
EOI	D9	Fim da imagem

Tabela 6.1 – Marcas de segmento mais comuns em ficheiros do formato JFIF.

Segmento APP0

Este segmento, que é único num ficheiro em formato JFIF, deve ser colocado imediatamente a seguir ao segmento SOI inicial. A informação contida num segmento APP0 é

Campo	**Bytes**	**Conteúdo**
APP0 length	2	Número de bytes contidos no segmento, incluindo os dois bytes deste campo.
Identifier	5	Este campo deve conter a cadeia “JFIF\0” em ASCII.
Version	2	Representação do número de versão (primeiro byte) e do número de revisão (segundo byte). Exemplo: a versão 1.02 é identificada pelo valor hexadecimal 0102.
Units for X and Y densities	1	Especifica a unidade de comprimento empregue nos valores de densidade vertical e horizontal dos píxeis. Pode assumir os valores 0 (não são especificadas unidades), 1 (píxeis por polegada) ou

		2 (píxeis por metro).
X density	2	Densidade horizontal dos píxeis.
Y density	2	Densidade vertical dos píxeis.
X thumbnail	1	Largura, em píxeis, do *thumbnail*[31] da imagem contida no ficheiro.
Y thumbnail	1	Altura, em píxeis, do *thumbnail* da imagem contida no ficheiro.
(RGB)n	3n	Valores RGB compactados em 24 bits, dos píxeis do *thumbnail*[32] da imagem contida no ficheiro, em que n=Xthumbnail×Ythumbnail.

Segmento DQT

Um ou mais segmentos DQT devem ser inseridos no ficheiro JFIF logo a seguir aos segmentos APPn, se existirem, ou ao segmento APP0. Um segmento DQT contém os valores de uma das tabelas empregues na quantização da cor realizada durante o processamento do conteúdo da imagem, antes da fase de compressão.

Os campos de um segmento DQT são:

Campo	**Bytes**	**Conteúdo**
DQT length	2	Número de bytes contidos no segmento, incluindo os dois bytes deste comprimento.
Table number	1	Indica qual das componentes de cor (Y, Cb ou Cr) empregará esta tabela.
Table values	n	Cada um dos n (n=DQTlength-3) bytes deste campo contém um elemento da tabela de quantização. O valor de n é, usualmente, 64.

Segmento SOF0

Este segmento descreve os principais parâmetros da imagem. Os seus dois bytes iniciais contêm o valor hexadecimal FFC0 e os seus campos são:

Campo	**Bytes**	**Conteúdo**
SOF0 length	2	Número de bytes contidos no segmento, incluindo os dois bytes deste comprimento.
Bit precision	1	Número de bits ocupados por cada componente de cor na imagem original, normalmente 8.
Image height	2	Altura da imagem, em píxeis.

[31] Um *thumbnail* é uma imagem reduzida da imagem contida no ficheiro, normalmente com dimensões iguais ou inferiores a 32×32 píxeis, e que pode desempenhar a função de ícone da imagem.

[32] Se Xthumbnail ou Ythumbnail forem nulos, não existe thumbnail no ficheiro e o campo (RGB)n não é incluído no segmento.

Image width	2	Largura da imagem, em píxeis.
Component number	1	Número de componentes de cor empregues pela imagem, obrigatoriamente 3.
Component Info	9	Contém três bytes por cada uma das componentes da cor, com o valor hexadecimal IIHVTT, em que: II – Identificador da componente (1, 2 ou 3) H – factor de amostragem horizontal da componente de cor V – factor de amostragem vertical da componente de cor TT – número da tabela de quantização empregue para esta componente de cor (ver segmento DQT)

Segmento DHT

Cada segmento DHT, identificado pelos seus dois bytes iniciais contendo o valor hexadecimal FFC4, contém uma das tabelas de Huffman empregues na compressão do conteúdo da imagem.

Os campos de um segmento DHT são:

Campo	**Bytes**	**Conteúdo**
DHT length	2	Número de bytes contidos no segmento, incluindo os dois bytes deste comprimento.
Type and table	1	Byte com o valor TN, em que T é o tipo de tabela (AC se T=1 ou DC se T≠1) e N o número da tabela.
Mark	16	Bytes contendo máscaras para os valores definidos pela tabela. A sua soma indica quantos valores da tabela estão armazenados no campo Values.
Values	n	Valores a introduzir na tabela de Huffman descrita por este segmento.

Segmento SOS

Este tipo de segmento introduz o segmento onde se encontra o conteúdo (comprimido) da imagem. O valor hexadecimal do identificador de um segmento SOS é FFDA.

Os campos da informação contida num segmento SOS são:

Campo	**Bytes**	**Conteúdo**
SOS length	2	Número de bytes contidos no segmento, incluindo os dois bytes deste comprimento.
Number of components	1	Número de componentes de cor da imagem (n), usualmente 3.

Component ID and Huffman tables	2n	Este conjunto de 2n bytes associa 2 bytes com o valor hexadecimal IIDA a cada uma das componentes de cor, sendo II – identificador da componente de cor (1-Y, 2-Cb, 3-Cr) D – Número da tabela DC de Huffman a ser empregue para descomprimir esta componente A – Número da tabela AC de Huffman a ser empregue para descomprimir esta componente
Padding	length-3-2n	Bytes de preenchimento do restante do bloco, se tal for necessário.

Imagem Comprimida

O conteúdo comprimido da imagem é colocado imediatamente a seguir ao segmento SOS. Este conteúdo encontra-se separado por componentes e, na sua leitura, ter-se-á que ter o cuidado de verificar se o byte seguinte a um byte com o valor FF contém ou não 0. Se assim for, o byte contendo o valor FF pertence à cadeia comprimida da imagem e o segundo byte é descartado. Se isto não suceder, então estamos perante um início de um novo segmento que, como tal, deverá ser processado.

7 Formatos de Imagem para Aplicações em Medicina

Existe um grande número de formatos de imagem empregues em aplicações de Medicina. Estes formatos variam de fabricante para fabricante de equipamentos médicos e, muitas vezes, de geração para geração de equipamentos de um mesmo fabricante. Alguns destes formatos baseiam-se em extensões a formatos de carácter geral adaptados ao tipo de aplicações e aos equipamentos. Para além dos dados das imagens, os formatos empregues em Medicina apresentam geralmente enormes quantidades de informação, agrupada em blocos de comprimento e estrutura muito variáveis. Esta informação descreve dados como a identificação do paciente e respectivos dados clínicos, datas e horas dos exames médicos, tipo de exames médicos, técnicas empregues nesses exames, sequência dos cortes (principalmente em TAC[33] e IRM[34]) e seriação de imagens[35].

As imagens contidas nestes formatos estão normalmente localizadas na parte final dos respectivos ficheiros, depois de toda a informação sobre o paciente e sobre o exame ou exames realizados. Nesta secção iremos apresentar, a título de exemplo, um formato de imagem empregue para armazenar imagens produzidas num equipamento de IRM.

[33] Tomografia Axial Computorizada.
[34] Imagem por Ressonância Magnética.
[35] Durante um exame TAC, é normalmente realizada uma série de imagens correspondentes a secções da zona anatómica a ser examinada. O conjunto de secções de um exame permite reconstituir um modelo tridimensional da zona examinada.

7.1 O formato GE Sygna

O formato GE Sygna é um dos muitos formatos de imagens empregues em aplicações de Medicina e, também, um dos muitos formatos utilizado em equipamentos de TAC e IRM da General Electric[36] (GE).

Este formato permite guardar imagens em tons de cinzento com uma profundidade máxima de 16 bits por píxel, ou seja 65536 níveis de intensidade, com ou sem compressão. O conteúdo das imagens encontra-se na parte final do ficheiro. Este conteúdo encontra-se organizado por varrimento vertical das linhas de cima para baixo e, dentro de cada linha, da esquerda para a direita.

No início do ficheiro existe um bloco de controlo que descreve o conteúdo do ficheiro e contém os parâmetros da imagem nele guardada. Os campos destes parâmetros são:

Campo	**Bytes**	**Conteúdo**
Identificador	4	Identificador do formato da imagem, a cadeia de caracteres "IMGF", em ASCII.
Offset da imagem	4	Posição (em bytes) do primeiro byte do conteúdo da imagem relativamente ao início do ficheiro.
Largura	4	Largura da imagem, em píxeis.
Altura	4	Altura da imagem em píxeis.
Profundidade	4	Número de bits por píxel, usualmente 16.
Compressão	4	Tipo de compressão aplicada ao conteúdo da imagem armazenada pelo ficheiro.

Além da informação referente aos parâmetros da imagem, o bloco de controlo pode conter outra informação referente à localização e comprimento de outros blocos de informação existentes no ficheiro. Os tipos de blocos que podem ser encontrados num ficheiro deste tipo são:

- Identificador (único) da imagem
- Bloco de compactação
- Bloco auxiliar, contendo informação necessária para descomprimir imagens comprimidas
- Bloco do histograma da imagem, contendo dados estatísticos sobre o emprego efectivo dos níveis de intensidade utilizados pela imagem
- Bloco descritor do plano de texto. O plano de texto destina-se a registar em placa fotográfica informação sobre o paciente e exame realizado
- Bloco descritor da base de dados, para associar a imagem ao conjunto de imagens que permitem descrever os volumes varridos por exames TAC ou IRM
- Bloco de dados introduzidos pelo operador, normalmente empregue para anexar comentários e anotações à imagem

[36] Este formato tem a curiosidade de ter sido empregue pela Universidade do Colorado em Denver ao colaborar no Visible Human Project do NIH.

- Bloco do conjunto de imagens, que, entre outra informação, contém informação sobre o equipamento empregue no exame realizado
- Bloco do exame médico, contendo informação sobre o tipo de exame médico, data e local onde foi realizado, identificação do paciente e outros dados como idade e sexo do paciente
- Bloco descrevendo a série de imagens a que a imagem pertence, contendo referências anatómicas à zona a que foi realizado o exame e ao tipo de protocolo de varrimento empregue na obtenção das imagens do exame
- Bloco da imagem, contendo informação sobre as características físicas (dimensões reais), resolução da imagem e outra informação referente à zona do paciente a que foi realizado o exame

Alguns destes blocos podem estar ou não presentes no ficheiro. A ausência de um bloco é assinalada por um comprimento nulo do bloco.

É prática corrente em aplicações médicas imprimir as imagens em chapas fotográficas para posterior exame pelos clínicos da mesma forma que examinam chapas radiológicas. Este tipo de meio permite representar um número de intensidades de tons de cinzento maior do 256 e, por esta razão, cada píxel da imagem tem a profundidade de 16 bits.

Para visualizar este tipo de imagens em monitores normais é necessário proceder à redução do número de níveis de intensidade. Uma forma expedita de o fazer consiste em converter a imagem para o formato PGM (Portable GrayMap) na sua variante em ASCII (tipo P2), declarando no prólogo da imagem o maior valor de intensidade da imagem[37] como a maior intensidade nela encontrado. É claro que obteremos assim um ficheiro de comprimento extremamente elevado. Para obviar a este inconveniente, poder-se-á, alternativamente, criar um ficheiro PGM de tipo binário (P5), fazendo corresponder níveis de cinzento compreendidos entre 0 e 255 aos níveis de cinzento entre 0 e o nível máximo presente na imagem original.

8 Comparação de Formatos

Os formatos de imagem descritos neste capítulo possuem características e funcionalidade muito variadas que a tabela 8.1 apresenta. O número destas características e os valores que podem assumir confundem o utilizador principiante ou médio que pretende seleccionar um formato de imagem. Normalmente, o utilizador está preocupado com dois aspectos: o tamanho dos ficheiros e a qualidade das imagens finais. Mas a natureza destas pode ser muito variada e, não raro, o utilizador é confrontado com resultados inesperados e, por vezes, desastrosos.

Apresentamos de seguida dois casos de imagens digitais que consubstanciam dois casos extremos:

- Imagens de conteúdo com qualidade fotográfica
- Imagens contendo gráficos simples

[37] Um conversor de formato de imagem poderá ser subsequentemente empregue para converter o ficheiro do formato PGM para outro formato em que a imagem ocupe menos espaço.

	PBM	DIB	GIF	PNG	JFIF
Número de cores suportadas	2, 16.777.216	2, 16, 256, 16.777.216	2, 4, 8, 16, 32, 64, 128, 256	2, 4, 16, 256, 16.777.216, 2^{48}	16.777.216
Tipo de cor	RGB	RGB	RGB	RGB	YCbCr
Compressão	não	RLE a 4 e 8 bits/píxel	LZW (s/perda)	LZ77 mod. (s/perda)	JPEG (c/perda)
Mapa de cores	não	só c/ 2, 16 e 256 cores	obrigatório em todos os tipos	só c/ 2, 4, 16, 256 cores	não
Entrelaçamento	não	não	opcional, por linhas	opcional, por píxeis	não
Varrimento vertical	descendente	ascendente	descendente	descendente	blocos
Transparência	não	não	Sim, 1 cor	Sim, plano alfa	não
Extensões convencionais	.pbm, .pgm, .ppm	.bmp	.gif	.png	.jpg
Outras características			canal GIF c/sequência controlável	canal PNG c/sequência controlável; corr. gama; CRC; network byte order	

Tabela 8.1 – Quadro comparativo das capacidades de vários formatos de imagem.

comparando o tamanho dos respectivos ficheiros quando convertidos para os formatos GIF, PNG, JFIF e PPM, a partir de imagens em formato DIB contendo toda a informação das imagens originais.

A imagem de qualidade fotográfica da figura 8.1 é uma imagem de 512×512 píxeis que emprega 230.426 cores. Os valores do tamanho dos ficheiros correspondentes a esta imagem[38], depois de convertida para vários formatos que a tabela 8.2 apresenta, permitem verificar que não existem diferenças significativas entre os formatos DIB e PPM, tal como era de esperar. Os ficheiros em formato PNG, embora sejam menores, não apresentam no entanto uma redução significativa do seu tamanho em relação àqueles formatos dado que o respectivo algoritmo de compressão não é adequado a uma imagem desta natureza, mesmo quando empregam filtragem adaptativa.

[38] Os valores apresentados na tabela dependem, como é óbvio, da imagem empregue. No entanto, valores obtidos com outras imagens de teste muito divulgadas (Lenna, por exemplo) não alteram a natureza das conclusões que aqui são apresentadas.

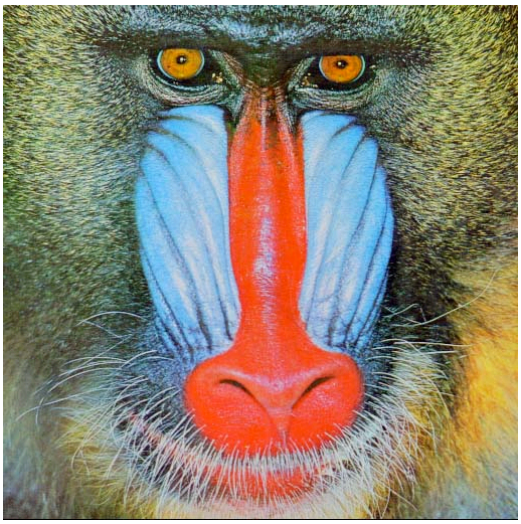


Figura 8.1 – Imagem de qualidade fotográfica

A redução significativa do tamanho dos ficheiros permitida pelo formato GIF é acompanhada pela inevitável redução do número de cores devido ao limite ao número de cores imposto pelo formato. A consequência é uma nítida redução da qualidade da imagem para um nível que deixa de ser aceitável.

Isto é bem visível no exemplo da figura 8.2 que apresenta em ampliação uma área de uma imagem cuja qualidade foi bastante degradada devido à redução do número de cores implícita na conversão da imagem para o formato GIF.

Os vários produtos empregues na conversão da imagem para o formato JFIF definem escalas de nível de qualidade cuja correspondência é difícil, senão impossível, de estabelecer. A tabela 8.3 apresenta essas escalas e os valores adoptados em cada um dos produtos para os níveis de teste da qualidade designados por máximo, médio e mínimo na conversão para formato JFIF.

	MS Photo Editor	Paint Shop Pro	Adobe Photoshop
DIB/BMP	786486	786486	786486
GIF s/entre	169369 (136)	261332 (256)	248878 (256)
GIF c/entre		266523 (256)	254103 (256)
PNG s/entre	755065	636923	781799
PNG c/entre		667648	788863
JFIF max	342808 (159067)	313125 (161704)	491285 (176062)
JFIF med	50717 (149903)	51151 (151270)	87798 (156760)
JFIF min	31768 (111645)	32300 (113683)	53248 (133424)
PPM binário		786475	
PPM ASCII		2893594	

Tabela 8.2 – Número de bytes ocupados pela imagem da Figura 8.1 em diferentes formatos produzidos por três produtos. A imagem original tem as dimensões de 512×512 píxeis e 230.426 cores (os números entre parêntesis assinalam o emprego pelo formato de um número de cores diferente deste).

Figura 8.2 – Ampliação de uma zona de uma imagem convertida para formato GIF com redução drástica do número de cores de que resulta uma área com variações abruptas de cor em vez de uma variação contínua.

A redução para cerca de metade do seu tamanho original em formato DIB dos ficheiros quando se empregou o formato JFIF com a qualidade máxima permitida está relacionada com a redução da quantidade de informação referente a cada píxel da imagem.

Com uma qualidade média, o número de cores reduz-se para pouco mais do que metade do número de cores original, mas os factores de compressão obtidos são da ordem de 10 ou mais e o comprimento dos ficheiros é entre 3 a 4 vezes mais pequeno do que os ficheiros GIF correspondentes, tudo isto sem que se detecte qualquer diferença entre a imagem original e a imagem comprimida.

Os ficheiros JFIF correspondentes à qualidade mínima são ainda mais pequenos mas, neste caso, já é possível detectar algumas diferenças entre a imagem original e as imagens comprimidas, excepto no caso da imagem produzida pelo Adobe Photoshop cuja compressão não é tão grande como a obtida pelos outros produtos. Mesmo assim, verifica-se que as imagens JFIF de pior qualidade são de qualidade muito superior às imagens em formato GIF.

Produto (escala)	MS Photo Editor (1-100)	Paint Shop Pró (99-1)	Photoshop (0-12)
Nível de qualidade			
Máximo	100	1	12
Médio	50	50	5
Mínimo	25	75	2

Tabela 8.3 – Escala de nível de qualidade da compressão JPEG empregues por diferentes produtos e valores para esses produtos referentes à qualidade máxima, média e mínima empregues no estudo comparativo.

Os valores da tabela 8.2 mostram que uma imagem comprimida por produtos diferentes resulta, em geral, em ficheiros de tamanhos diferentes. Isto deve-se à forma diversa com que os produtos implementam os mesmos algoritmos de compressão. Da tabela também é possível concluir que o emprego de entrelaçamento tem o efeito de aumentar um pouco, embora não significativamente, o espaço ocupado pelos ficheiros que utilizam esta técnica.

A imagem da figura 8.3 situa-se no outro extremo do espectro de tipos de imagens relativamente às imagens de qualidade fotográfica. A imagem, com as dimensões de 717×494 píxeis, apresenta 10 cores e contém um gráfico simples em que, quando a cor varia, a variação de cor é abrupta de um píxel para o píxel contíguo. Na imagem existem grupos de dimensão apreciável de píxeis contíguos da mesma cor. A tabela 8.4 apresenta os comprimentos em bytes dos ficheiros resultantes da conversão da imagem da figura para vários formatos.

Tal como para a imagem de qualidade fotográfica, não existem diferenças significativas entre os formatos DIB e PPM no que respeita ao tamanho dos respectivos ficheiros. Mas os ficheiros resultantes da conversão da imagem para os formatos GIF e PNG apresentam agora tamanhos muito semelhantes. Se o número máximo de cores da imagem for reduzido de 16.777.216 para 256 antes da conversão para o formato PNG, os ficheiros obtidos serão nitidamente mais pequenos que os ficheiros da mesma imagem em formato GIF.

A conversão desta imagem para o formato JFIF resulta sempre em ficheiros de comprimento maior do que os dos formatos GIF ou PNG. Mas, mais importante, esta conversão é acompanhada por uma nítida degradação da imagem. Com efeito, mesmo empregando o formato JFIF no maior nível de qualidade que é possível, as imagens convertidas por qualquer dos produtos apresentam artefactos devidos a aliasing. Este efeito acentua-se ainda mais para as imagens JFIF correspondentes aos níveis de qualidade média e mínima, sendo os resultados particularmente catastróficos neste último caso. A degradação da qualidade das imagens é muito nítida e é acompanhada pela multiplicação incontrolada do número de cores efectivamente empregues nas imagens, com cores e linhas muito esborratadas.

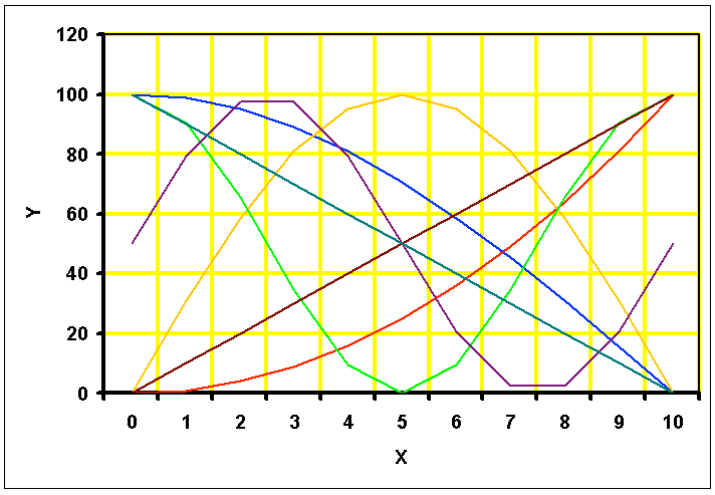


Figura 8.3 – Imagem contendo um gráfico simples

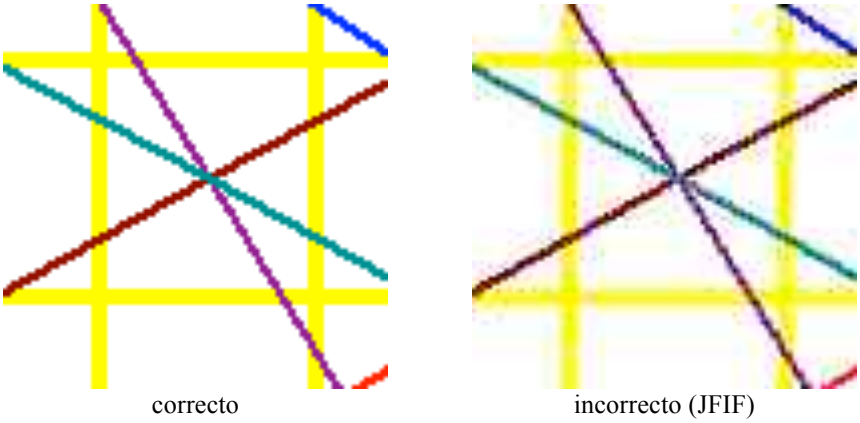
	MS Photo Editor	Paint Shop Pro	Adobe Photoshop
DIB/BMP	1063142	1063142	1063142
GIF s/entre	19271 (18)	17581	16268
GIF c/entre		17904	16667
PNG s/entre	18951	19097	29861
PNG c/entre		28482	39103
PNG 256 cores	14795 (18)	12899	14975 (18)
JFIF max	174285 (111)	161770 (2764)	226607 (444)
JFIF med	44330 (27550)	44262 (28097)	73789 (21443)
JFIF min	31768 (29756)	31495 (30223)	54583 (26423)
PPM binário		1062637	
PPM ASCII		4161787	

Tabela 8.4 - Número de bytes ocupados pela imagem da Figura 8.3 em diferentes formatos produzidos por três produtos. A imagem original tem as dimensões de 717×494 píxeis e 10 cores (os números entre parêntesis assinalam o emprego pelo formato de um número de cores diferente).

A figura 8.4 permite apreciar em ampliação os artefactos introduzidos na imagem da figura 8.3 na sua conversão para o formato JFIF.

Na figura 8.5 é possível apreciar a degradação progressiva da qualidade da imagem da figura 2.2 quando esta é convertida para o formato JFIF com cada vez menor nível de qualidade.

Os exemplos anteriores mostram que o formato JFIF é o formato ideal para imagens de qualidade fotográfica, mas que não deve ser empregue com imagens com pequeno número de cores ou com variações súbitas de cor entre píxeis contíguos. Para este caso os formatos GIF ou PNG são muito mais apropriados e, se o número de cores for inferior a 256 cores, é conveniente reduzir primeiro a profundidade da imagem para 256 cores e, em seguida, convertê-la para o formato PNG.



correcto incorrecto (JFIF)

Figura 8.4 – Área ampliada da imagem da figura 8.3 mostrando os artefactos introduzidos pela conversão da imagem para o formato JFIF.

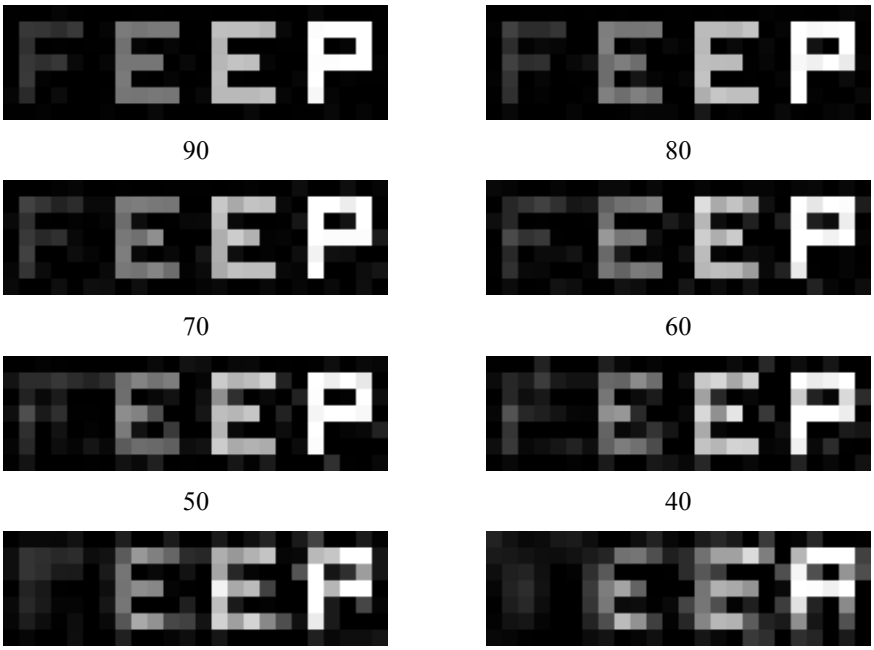


Figura 8.5 – Degradação progressiva da qualidade de uma imagem para a qual o formato JFIF é totalmente inapropriado. Os níveis de qualidade de cada imagem correspondem a uma escala com nível máximo de qualidade igual a 100 e um nível de qualidade mínima de 1.

O emprego do formato JFIF deve ser feito com alguns cuidados. Em primeiro lugar há que ter cuidado com o nível de redução de qualidade a empregar, pois a sua influência na qualidade dos resultados varia de imagem para imagem e, também, com as dimensões da imagem, sendo conveniente realizar algumas experiências antes de optar por um determinado nível. Um segundo aspecto, talvez mais importante do que o primeiro, consiste no cuidado a ter quando se reduzem as dimensões de uma imagem em formato JFIF. A redução das dimensões de imagens que já tenham sofrido perdas significativas resulta em imagens de muito má qualidade porque a informação necessária ao processo de redução das dimensões é insuficiente. Neste caso, a forma correcta de proceder consiste em, a partir de uma imagem JFIF com a máxima qualidade possível, reduzir primeiro as suas dimensões e, só depois, comprimi-la com perda.

Exercícios

1. Explique como foram calculados os tempos ao fim dos quais se encontra completa cada uma das passagens do algoritmo de entrelaçamento da figura 1.3.
2. Faça o mesmo para o caso da imagem da figura 1.4. Com base nos resultados que obteve, explique a vantagem do algoritmo entrelaçamento Adam 7 em relação ao algoritmo de entrelaçamento empregue pelo formato GIF.
3. A taxa de transmissão efectiva que se verifica durante o carregamento de uma imagem GIF, com as dimensões de 320×240 píxeis, por um navegador da WWW é de 5.500 bit/s. Calcule os tempos ao fim dos quais as sucessivas passagens de entrelaçamento se encontram completas.
4. Explique por que razão, para uma imagem do tipo True Color, não tem vantagem o emprego de um mapa de cores.
5. Se, numa imagem do tipo High Color de m×n píxeis em que cada componente de cor ocupa 1 byte, for empregue um mapa de cores em que o número de cores possa ser inferior ao número limite (65.536 cores), determine o número de cores para o qual o emprego do mapa de cores resulta numa representação da imagem com o mesmo comprimento que se obtém quando o mapa não é empregue.
6. Para o resultado do exercício anterior, explique o que sucederia para valores do número de cores superiores e inferiores ao valor que determinou.
7. Para uma imagem do tipo High Color de m×n píxeis em que cada componente de cor ocupa 1 byte, qual é o tamanho da imagem a partir do qual o emprego de um mapa de cores contendo 65536 cores deixa de ter interesse?
8. Calcule o factor de compressão mínimo permitido pelo formato JFIF para uma imagem do tipo True Color quando o modelo de cor YCbCr, usado por este formato, emprega 8 bits para representar a luminância e 4 bits para representar as componentes de cromaticidade do modelo.
9. Determine o número de bytes ocupado por uma imagem de 200×150 píxeis, com 8 bits por píxel, em que a metade superior é verde e a inferior amarela, quando a imagem é comprimida pelo algoritmo de compressão RLE. Compare o resultado obtido com o comprimento da imagem não comprimida.
10. Compare os resultados que obteve no exercício anterior com os que obteria se a imagem fosse constituída por uma metade esquerda de cor verde e uma metade direita de cor amarela. Explique a razão das diferenças que encontrar.
11. Explique as vantagens e desvantagens do emprego da máscara para obter efeitos de transparência empregue pelo formato PNG em relação à transparência que é possível obter com o formato GIF com base numa dada cor.