Procesamiento de Imágenes: Estructura de Archivos BMP

Ing. Agustín Cruz Contreras,
M. en C. Juan Carlos González Robles,
M. en C. Juan Carlos Herrera Lozada
Profesores del CIDETEC-IPN.

El presente trabajo describe la forma de interpretar la información contenida en archivos BMP, con el propósito de aplicar algoritmos propios de procesamiento a imágenes capturadas o generadas con sistemas comerciales.

Introducción

Las imágenes se capturan o generan a través de equipos y programas comerciales, empleando formatos propios, como es el caso de Photoshop y Corel, o formatos con mayor difusión como BMP, JPG, PCX, etc.

En el desarrollo e implementación de algoritmos para el procesamiento de imágenes, se requiere la matriz de datos que contiene el color para cada pixel, esto no se tiene de manera directa si la imagen está almacenada con algún formato.

Una imagen puede almacenarse en un archivo siguiendo diferentes formatos, utilizando muchos de ellos compresión de datos. Cada uno tiene sus ventajas y desventajas, pero todos ellos tienen algunas características en común: Siempre se utiliza una cabecera en el archivo que identifica el tipo de formato del que se trata, conteniendo información necesaria para interpretar el archivo (el tamaño de la imagen o el número de colores, etc.), después de la cabecera se encuentran los datos de la imagen, generalmente comprimidos con un algoritmo específico.

La imagen puede tener más o menos colores; entre más colores, será necesario un mayor número de bits por pixel para indicar el color del que se trata. Cuantos más colores, mejor calidad tendrá la imagen, pero de mayor tamaño será el archivo. Normalmente el número de colores es 16, 256 o 16 millones, lo que requiere 4, 16 o 24 bits por pixel. En el caso de utilizar 16 o 256 colores, debe especificarse a que color real corresponde cada uno de esos colores, es decir, que cantidades de Rojo, Verde y Azul serán utilizadas para representar el color en la pantalla. La tabla que asocia a cada color con las correspondientes cantidades de Rojo, Verde y Azul se llama paleta de colores. Puede ser modificada en función de la imagen, por lo que es necesario guardarla en el archivo. En el caso particular de utilizar 16 millones de colores, no se utiliza paleta de colores, pues la relación entre número de color y cantidad de Rojo, Verde y Azul es implícita: De los 24 bits por pixel, se utilizan 8 por color.

Algunos Formatos de Imagen

Con la intención de dar una idea clara sobre los formatos de imagen, a continuación se describen brevemente algunos de éstos.

BMP

El formato BMP (Windows BitMaP) es probablemente el más simple que existe, y consiste en una cabecera seguida por los valores de cada pixel, comenzando por la línea inferior y terminando por la superior, pixel a pixel de izquierda a derecha, manteniendo las líneas un tamaño múltiplo de 32. Su única ventaja es su sencillez. Su gran desventaja es el enorme tamaño de los archivos. Este formato se puede considerar como universal dado que prácticamente todas las aplicaciones lo pueden manejar.

PCX

En el formato PCX (de PC Paintbrush), los datos están comprimidos. Suponiendo que estemos utilizando 256 colores (un byte por pixel), el algoritmo consiste simplemente en reemplazar las secuencias de N pixels consecutivos del mismo color por dos bytes, de forma que el primero indique el número N de repeticiones y el segundo indique el color. Este algoritmo permite reducir el tamaño del archivo cuando la imagen sea un dibujo con planos de color constante, pues serán muchos los pixels consecutivos del mismo color. Sin embargo cuando haya algún pixel aislado de un color,

éste se reemplazará por los dos bytes. Por ese motivo es posible que en algunos casos llegue a aumentar el tamaño del archivo. Esto suele pasar con imágenes escaneadas, pues es improbable que varios pixels consecutivos tengan exactamente el mismo color.

GIF

El formato GIF (Graphic Interchange Format) fue inventado por Compuserve. Utiliza un algoritmo de compresión similar al que usan los programas de compresión convencionales. Consiste en no detectar sólo las repeticiones de un color, sino en detectar las repeticiones de ciertas secuencias, mediante un diccionario que se va construyendo. Al igual que el formato PCX, funciona especialmente bien con imagenes en los que muchos pixels consecutivos tienen el mismo color, o se repiten secuencias de colores, pero también funciona bien con fotografías escaneadas o cualquier otra imagen. Además permite definir un color transparente, por lo que es muy útil para páginas Web.

JPG

El formato JPG utiliza un complejo algoritmo de compresión con pérdida de información. Es decir, el algoritmo modifica ligeramente los datos de forma que la imagen puede comprimirse mucho más. Gracias a esto el formato JPG es uno de los que más comprimen. Las modificaciones realizadas son inapreciables si se trata de una fotografía escaneada. Sin embargo, si se trata de un dibujo aparecen pixels visibles en la imagen. Por esta razón este formato es muy útil para fotografías escaneadas, pero no para dibujos, y por eso no permite definir colores como transparentes.

Estructura de un archivo BMP

La estructura de un archivo BMP comprende las siguientes secciones:

- Encabezado
- Paleta de colores
- Mapa de bits

Con la intención de abordar el caso más simple se optó por manejar imágenes a 24 bits por color, con lo cual no se requiere de paleta de colores, y la información del color está dada por la combinación de rojo, verde y azul, en valores de 0 a 255.

La cabecera se integra por los siguientes campos:

```
struct header{
  char type[2];              2 bytes con los caracteres "BM"
  unsigned long size;        4 bytes que indican el tamaño total del archivo
  char reserved[4];          4 bytes reservados
  unsigned long offset;      4 bytes que indican el offset desde el comienzo
                             del archivo hasta el inicio del mapa de bits.
          };

struct bmp_info{
  unsigned long bytes_in_header;   4 bytes reservados
  unsigned long widht;             4 bytes que indican el ancho del gráfico
                                   (en pixels).
  unsigned long height;            4 bytes que indican el alto del gráfico (en pixels)
  int planes;                      2 bytes que indican la cantidad de planos del gráfico
  int bits_per_pixel;              2 bytes que indican la cantidad de bits por pixel

  unsigned long compression;       Los siguientes campos
  unsigned long size_image;        de los cuales podemos prescindir
  unsigned long h_resolution;      en nuestra aplicación.
  unsigned long v_resolution;
  unsigned long n_indexes;         formando un total de 54 bytes.
  unsigned long n_i_indexes;
```

Se debe tener presente que en el archivo los valores se organizan del byte menos significativo, al más significativo, y que la imagen se encuentra invertida, esto es: al inicio se encuentra la línea inferior, al final la línea superior, acomodadas de izquierda a derecha, con una longitud de línea en bytes múltiplo de 32. En caso de que la longitud de línea difiera de un múltiplo de 32, el programa de creación agrega los bytes faltantes para completar la longitud múltiplo de 32. En nuestro ejemplo la longitud de línea es de 10 pixels x 3=30 bytes, y el programa de creación agrega dos bytes con valor de cero para tener una longitud de 32 bytes.

Programa

El programa tiene como propósito extraer la información de la imagen, ubicando ésta en una matriz, que corresponde en dimensiones al alto y ancho de la imagen.

El programa se integra por los siguientes pasos:

- Apertura de archivo.
- Lectura de los campos "ancho y alto": Con estos se determina el tamaño de la matriz que se requiere para almacenar la imagen.
- Lectura del campo "tamaño de cabecera": A partir de aquí comienza la información de la imagen.
- Lectura de datos considerando la organización descrita anteriormente.
- Desplegado de imagen en pantalla: Para verificar el éxito de la lectura.
- Cierre de archivo.

Ejemplo de estudio: Utilizando el "Paint" de Windows se formó una imagen de prueba con la composición mostrada en la **figura 1** (página siguiente).

R	R	R	R	R	R	R	R	R	G
V	V	V	V	V	V	V	V	V	G
A	A	A	A	A	A	A	A	A	G
N	N	N	N	N	N	N	N	N	G
B	B	B	B	B	B	B	B	B	G

Figura 1. Imagen sintética de prueba.

Donde:

	R	G	B
R= Rojo	FF	00	00
V= Verde	00	FF	00
A= Azul	00	00	FF
N=Negro	00	00	00
B=Blanco	FF	FF	FF
G= Gris	7F	7F	7F

Realizando el volcado del archivo en hexadecimal se tiene la información de la **figura 2**.

A cada casilla le corresponde un byte de información, contando las casillas, se tienen 214 bytes, que es el tamaño de archivo.

Asociando la información dada con los datos del archivo tenemos el esquema de mostrado en la figura 3.

Todas líneas mantienen la misma estructura, por lo que se considera suficiente, mostrar sólo el detalle de la primera en el archivo.

42	4D	D6	00	00	00	00	00	00	00	36	00	00	00	28	00
00	00	0A	00	00	00	05	00	00	00	01	00	18	00	00	00
00	00	A0	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
FF	7F	7F	7F	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	7F	7F	7F	00	00	FF	00	00	FF	00	00	FF	00	00	FF
00	00	FF	00	00	FF	00	00	FF	00	00	FF	00	00	FF	00
00	7F	7F	7F	00	00	00	FF	00	00	FF	00	00	FF	00	00
FF	00	00	FF	00	00	FF	00	00	FF	00	00	FF	00	00	FF
00	7F	7F	7F	00	00	00	00	FF	00	00	FF	00	00	FF	00
00	FF	00	00	FF	00	00	FF	00	00	FF	00	00	FF	00	00
FF	7F	7F	7F	00	00										

Figura 2. Volcado de archivo de imagen de prueba

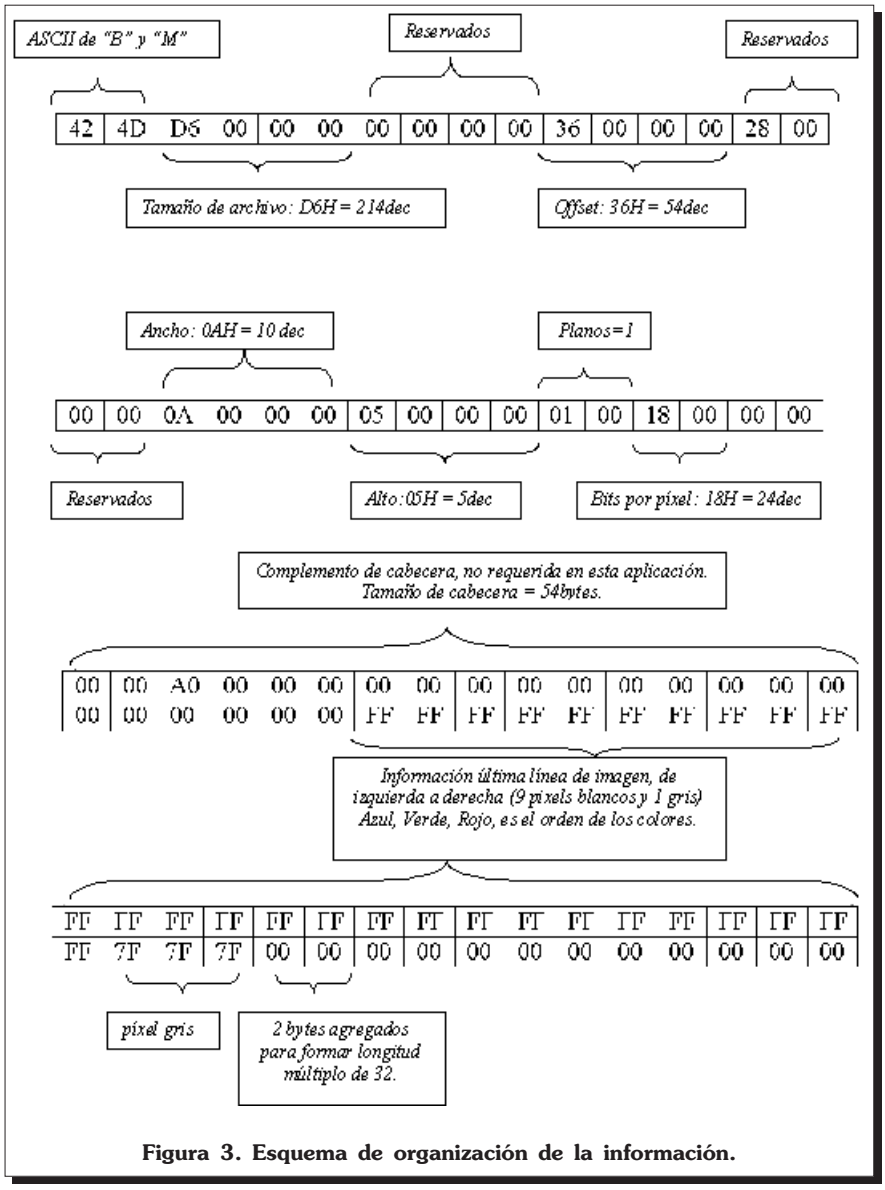

Figura 3. Esquema de organización de la información.

BIBLIOGRAFÍA

[1] Steve Rimmer. *"The Graphic File Toolkit"*. Addison Wesley. 1992.

[2] Steve Rimmer. *"Bit-Mapped Graphics"*. Mc Graw Hill. 1993.

[3] David C. Kay, John R. Levine. *"Graphics File Formats"*. Mc Graw Hill.1995.