Chapter 3
Graphics and Image Data Representations

3.1 Graphics/Image Data Types

- The number of file formats used in multimedia continues to proliferate. For example, Table 3.1 shows a list of some file formats used in the popular product Adobe Premiere.

Table 3.1: Adobe Premiere file formats.

Image	Sound	Video
BMP, GIF, JPG, EPS, PNG, PICT, PSD, TIF, TGA	AIFF, AAC, AC3, MP3, MPG, M4A, MOV, WMA	AVI, MOV, DV, FLV, MPG, WMA, WMV, SWF, M4V, MP4, MXF

→Link to details on Director file formats.

3.1.1 1-Bit Images

- Each pixel is stored as a single bit (0 or 1), so also referred to as **binary image**.
- Such an image is also called a 1-bit **monochrome** image since it contains no color.
- Fig. 3.1 shows a 1-bit monochrome image (called "Lena" by multimedia scientists — this is a standard image used to illustrate many algorithms).

Fig. 3.1: Monochrome 1-bit Lena image.

3.1.2 8-bit Gray-level Images

- Each pixel has a gray-value between 0 and 255. Each pixel is represented by a single byte; e.g., a dark pixel might have a value of 10, and a bright one might be 230.

- **Bitmap**: The two-dimensional array of pixel values that represents the graphics/image data.

- **Image resolution** refers to the number of pixels in a digital image (higher resolution always yields better quality).

  - Fairly high resolution for such an image might be 1600 x 1200, whereas lower resolution might be 640 x 480.

- **Frame buffer**: Hardware used to store bitmap.

  - **Video card** (actually a *graphics card*) is used for this purpose.

  - The resolution of the video card does not have to match the desired resolution of the image, but if not enough video card memory is available then the data has to be shifted around in RAM for display.

- An 8-bit image can be thought of as a set of 1-bit **bit-planes**, where each plane consists of a 1-bit contribution to the image : a bit is turned on if the image pixel value has a nonzero value for that bit. E.g., if a pixel stores 1 byte, then all the pixels with byte value > 127 have the most significant bit turned on.

- Fig. 3.2 displays the concept of bit-planes graphically.

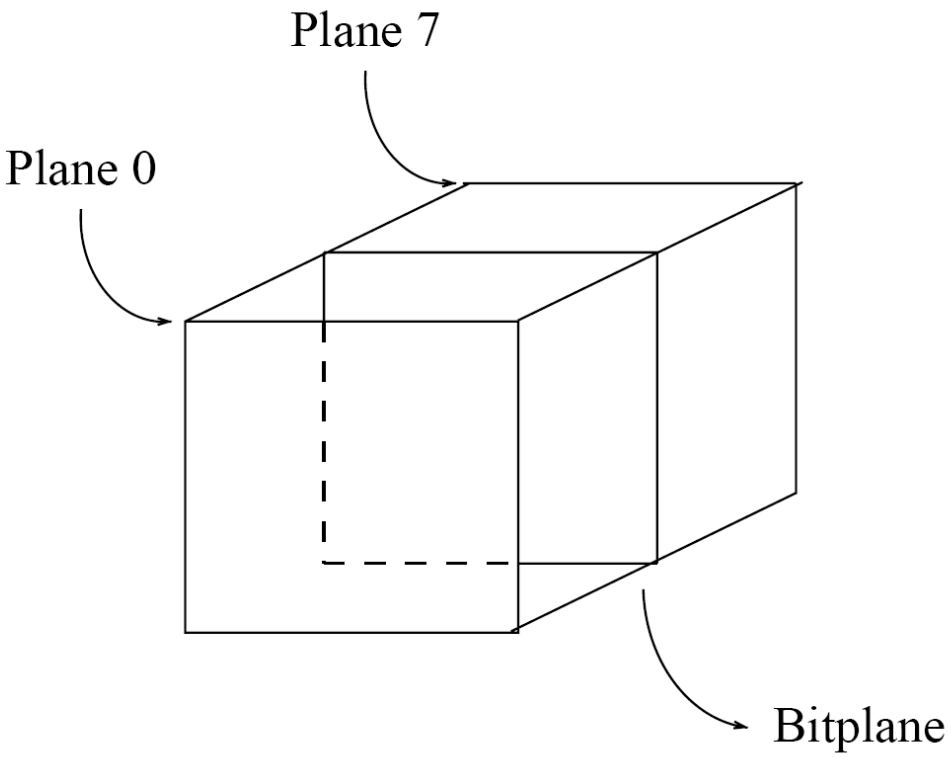

Fig. 3.2: Bit-planes for 8-bit grayscale image.

- Each pixel is usually stored as a byte (a value between 0 to 255), so a 640 x 480 grayscale image requires 300 kB of storage (640 x 480 = 307,200).

- Fig. 3.3 shows the Lena image again, but this time in grayscale.

- When an image is printed, the basic strategy of **dithering** is used, which trades intensity resolution for spatial resolution to provide ability to print multi-level images on 2-level (1-bit) printers.

Fig. 3.3: Grayscale image of Lena.

Dithering

- **Dithering** is used to calculate patterns of dots such that values from 0 to 255 correspond to patterns that are more and more filled at darker pixel values, for printing on a 1-bit printer.

- The main strategy is to replace a pixel value by a larger pattern, say 2 x 2 or 4 x 4, such that the number of printed dots approximates the varying-sized disks of ink used in analog, in **halftone printing** (e.g., for newspaper photos).

  1. Half-tone printing is an analog process that uses smaller or larger filled circles of black ink to represent shading, for newspaper printing.

  2. For example, if we use a 2 x 2 **dither matrix:**

$$\begin{pmatrix} 0 & 2 \\ 3 & 1 \end{pmatrix}$$

we can first re-map image values in 0..255 into the new range 0..4 by (integer) dividing by 256/5. Then, e.g., if the pixel value is 0 we print nothing, in a 2 x 2 area of printer output. But if the pixel value is 4 we print all four dots.

- The rule is:

  If the intensity is > the dither matrix entry then print an on dot at that entry location: replace each pixel by an *n x n* matrix of dots.

- Note that the image size may be much larger, for a dithered image, since replacing each pixel by a 4 x 4 array of dots, makes an image 16 times as large.

- A clever trick can get around this problem. Suppose we wish to use a larger, 4 x 4 dither matrix, such as

$$\begin{pmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{pmatrix}$$

- An **ordered dither** consists of turning on the printer out-put bit for a pixel if the intensity level is greater than the particular matrix element just at that pixel position.

- Fig. 3.4 (a) shows a grayscale image of "Lena". The ordered-dither version is shown as Fig. 3.4 (b), with a detail of Lena's right eye in Fig. 3.4 (c).

- An algorithm for ordered dither, with n x n dither matrix, is as follows:

Algorithm 3.1: Ordered Dither

```
1: for x = 0 to x_max do                    // columns
2:     for y = 0 to y_max do                // rows
3:         i = x mod n
4:         j = y mod n
5:         // I(x, y) is the input, O(x, y) is the output, D is the dither matrix.
6:         if I(x, y) > D(i, j) then
7:             O(x, y) = 1;
8:         else
9:             O(x, y) = 0;
```

(a) (b) (c)

Fig. 3.4: Dithering of grayscale images.

(a): 8-bit grey image "lenagray.bmp".

(b): Dithered version of the image.

(c): Detail of dithered version.

3.1.3 Image Data Types

- The most common data types for graphics and image file formats — 24-bit color and 8-bit color.
- Some formats are restricted to particular hardware / operating system platforms, while others are "cross-platform" formats.
- Even if some formats are not cross-platform, there are conversion applications that will recognize and translate formats from one system to another.
- Most image formats incorporate some variation of a **compression** technique due to the large storage size of image files. Compression techniques can be classified into either **lossless** or **lossy**.

3.1.4 24-bit Color Images

- In a color 24-bit image, each pixel is represented by three bytes, usually representing RGB.
  - This format supports 256 x 256 x 256 possible combined colors, or a total of 16,777,216 possible colors.
  - However such flexibility does result in a storage penalty: A 640 x 480 24-bit color image would require 921.6 kB of storage without any compression.
- **An important point**: many 24-bit color images are actually stored as 32-bit images, with the extra byte of data for each pixel used to store an *alpha* value representing special effect information (e.g., transparency).
- Fig. 3.5 shows the image **forestfire.bmp**, a 24-bit image in Microsoft Windows BMP format. Also shown are the grayscale images for just the Red, Green, and Blue channels, for this image.

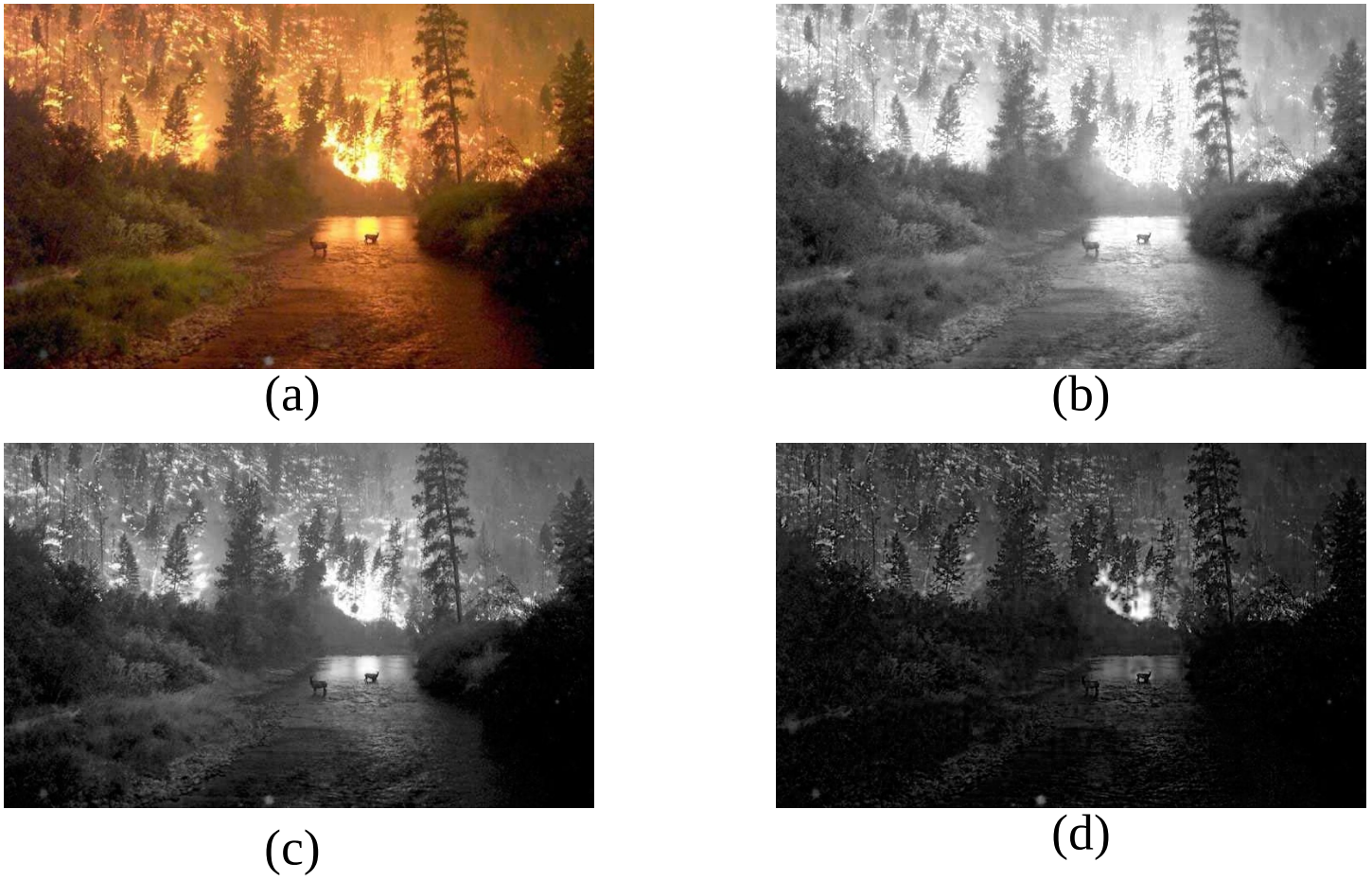

Fig. 3.5: High-resolution color and separate R, G, B color channel images. (a): Example of 24-bit color image "forestfire.bmp". (b, c, d): R, G, and B color channels for this image

3.1.5 Higher Bit-depth Images

More information about the scene being imaged can be gained by using more accuracy for pixel depth (64 bits, say); or by using special cameras that view more than just three colors (i.e., RGB).

- could use invisible light (e.g., infra-red, ultraviolet) for security cameras: "dark flash".
- use higher-dimensional medical images of skin (> 3-D) to diagnose skin carcinoma.
- in satellite imaging, use high-D to obtain types of crop growth, etc.

Such images are called *multispectral* (more than 3 colors) or hyperspectral (a great many image planes, say 224 colors for satellite imaging).

3.1.6 8-Bit Color Images

- Many systems can make use of 8 bits of color information (the so-called "256 colors") in producing a screen image.

- Such image files use the concept of a **lookup table** to store color information.
  - Basically, the image stores not color, but instead just a set of bytes, each of which is actually an index into a table with 3-byte values that specify the color for a pixel with that lookup table index.

- Fig. 3.6 shows a 3D histogram of the RGB values of the pixels in "forestfire.bmp".

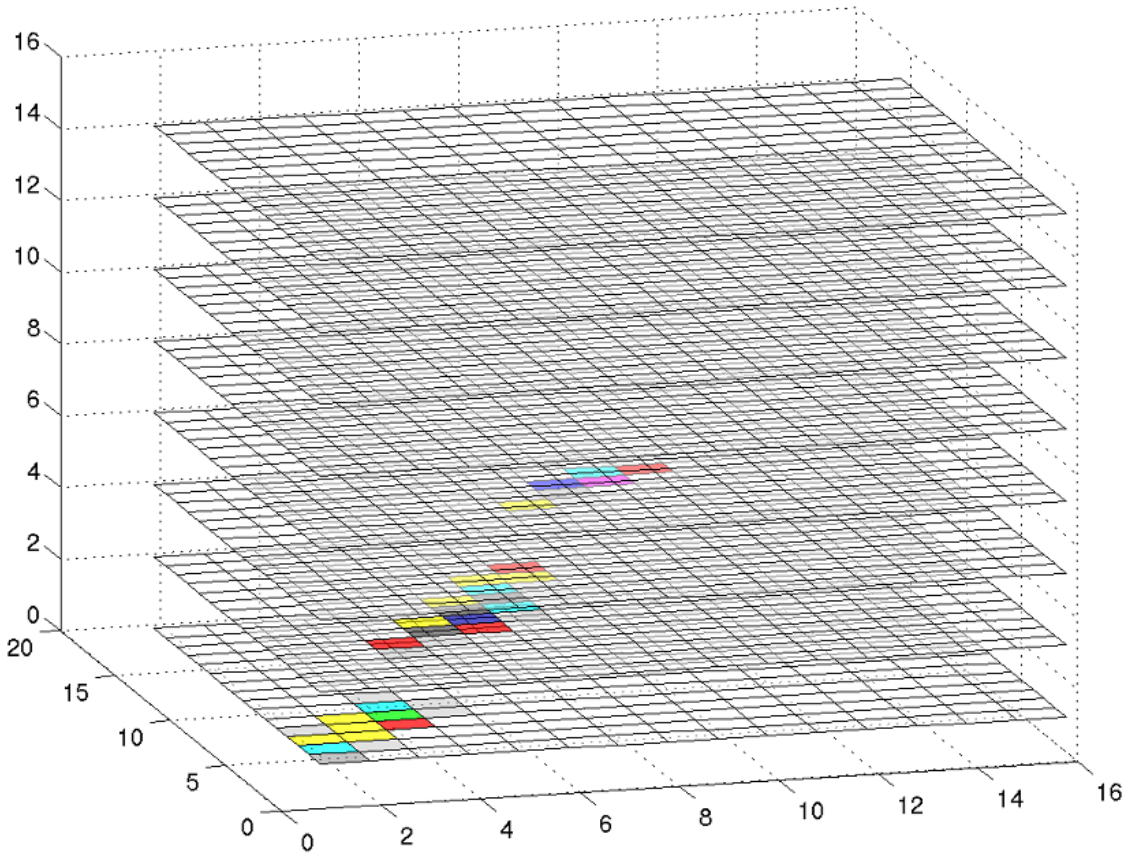

Fig. 3.6: 3-dimensional histogram of RGB colors in "forestfire.bmp".

- Fig. 3.7 shows the resulting 8-bit image, in GIF format.

Fig. 3.7: Example of 8-bit color image.

- Note the great savings in space for 8-bit images, over 24-bit ones: a 640 x 480 8-bit color image only requires 300 kB of storage, compared to 921.6 kB for a color image (again, without any compression applied).

3.1.7 Color Look-up Tables (LUTs)

- The idea used in 8-bit color images is to store only the index, or code value, for each pixel. Then, e.g., if a pixel stores the value 25, the meaning is to go to row 25 in a color look-up table (LUT).

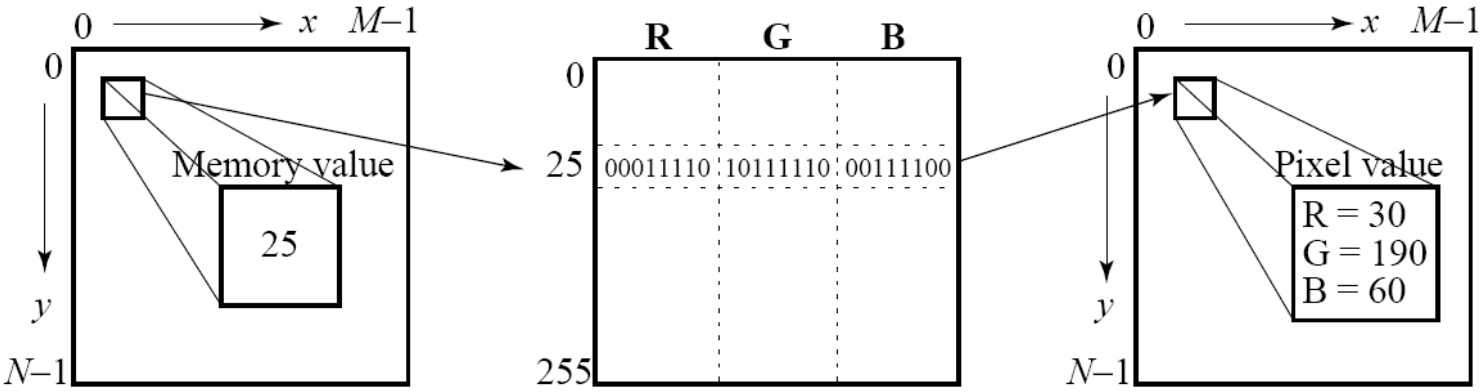

Fig. 3.8: Color LUT for 8-bit color images.

- A **Color-picker** consists of an array of fairly large blocks of color (or a semi-continuous range of colors) such that a mouse-click will select the color indicated.

  - In reality, a color-picker displays the palette colors associated with index values from 0 to 255.
  - Fig. 3.9 displays the concept of a color-picker: if the user selects the color block with index value 2, then the color meant is cyan, with RGB values (0, 255, 255).

- A very simple animation process is possible via simply changing the color table: this is called **color cycling** or **palette animation**.

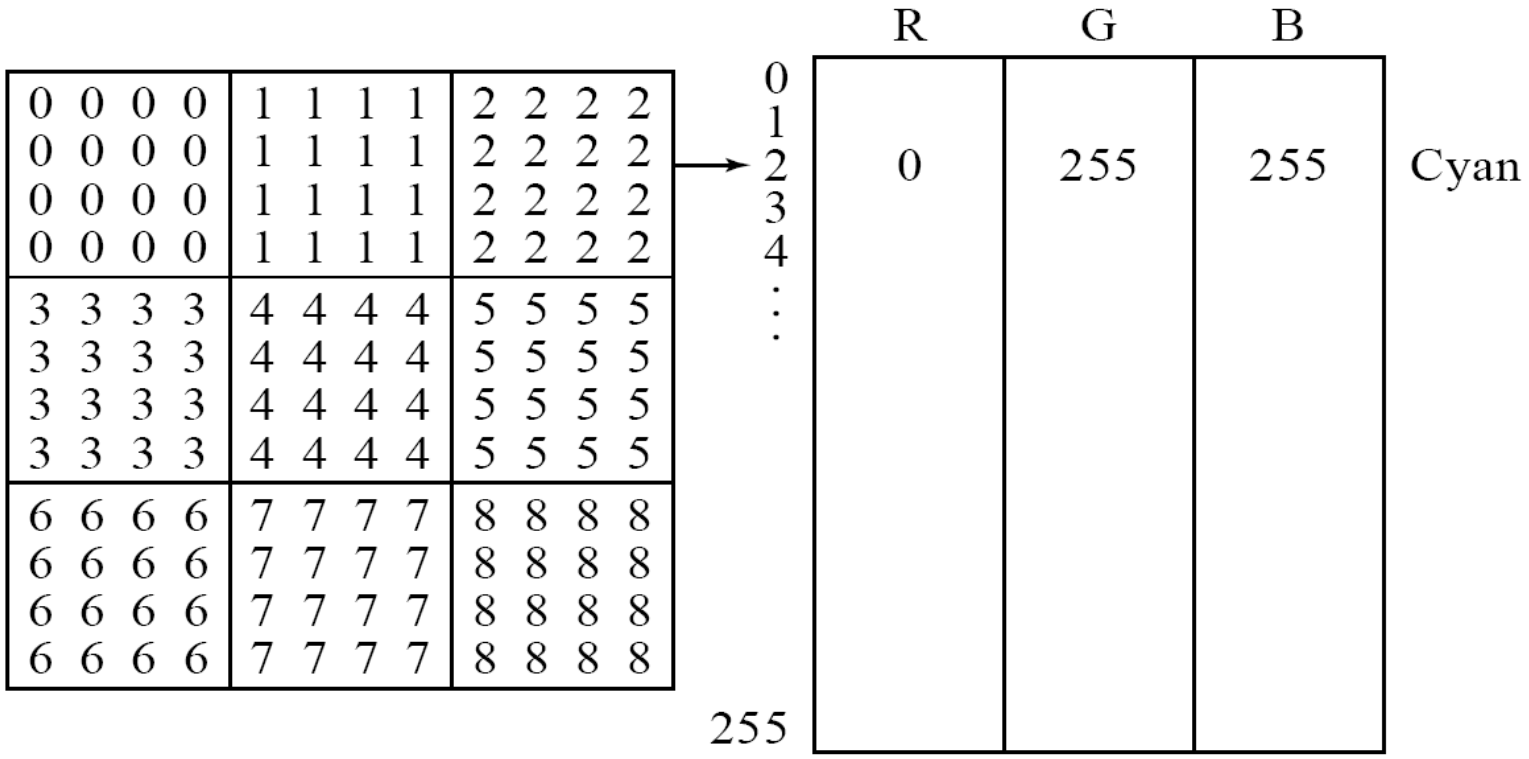

Fig. 3.9: Color-picker for 8-bit color: each block of the color-picker corresponds to one row of the color LUT

Fig. 3.10 (a) shows a 24-bit color image of "Lena", and Fig. 3.10 (b) shows the same image reduced to only 5 bits via dithering. A detail of the left eye is shown in Fig. 3.10 (c).

(a) (b) (c)

Fig. 3.10: (a): 24-bit color image "lena.bmp". (b): Version with color dithering. (c): Detail of dithered version.

How to devise a color look-up table

- The most straightforward way to make 8-bit look-up color out of 24-bit color would be to divide the RGB cube into equal slices in each dimension.

  a) The centers of each of the resulting cubes would serve as the entries in the color LUT, while simply scaling the RGB ranges 0..255 into the appropriate ranges would generate the 8-bit codes.

  b) Since humans are more sensitive to R and G than to B, we could shrink the R range and G range 0..255 into the 3-bit range 0..7 and shrink the B range down to the 2-bit range 0..3, thus making up a total of 8 bits.

  c) To shrink R and G, we could simply divide the R or G byte value by (256/8)=32 and then truncate. Then each pixel in the image gets replaced by its 8-bit index and the color LUT serves to generate 24-bit color.

- **Median-cut algorithm**: A simple alternate solution that does a better job for this color reduction problem.

a) The idea is to sort the R byte values and find their median; then values smaller than the median are labelled with a "0" bit and values larger than the median are labelled with a "1" bit.

b) This type of scheme will indeed concentrate bits where they most need to differentiate between high populations of close colors.

c) One can most easily visualize finding the median by using a histogram showing counts at position 0..255.

d) Fig. 3.11 shows a histogram of the R byte values for the `forestfire.bmp` image along with the median of these values, shown as a vertical line.

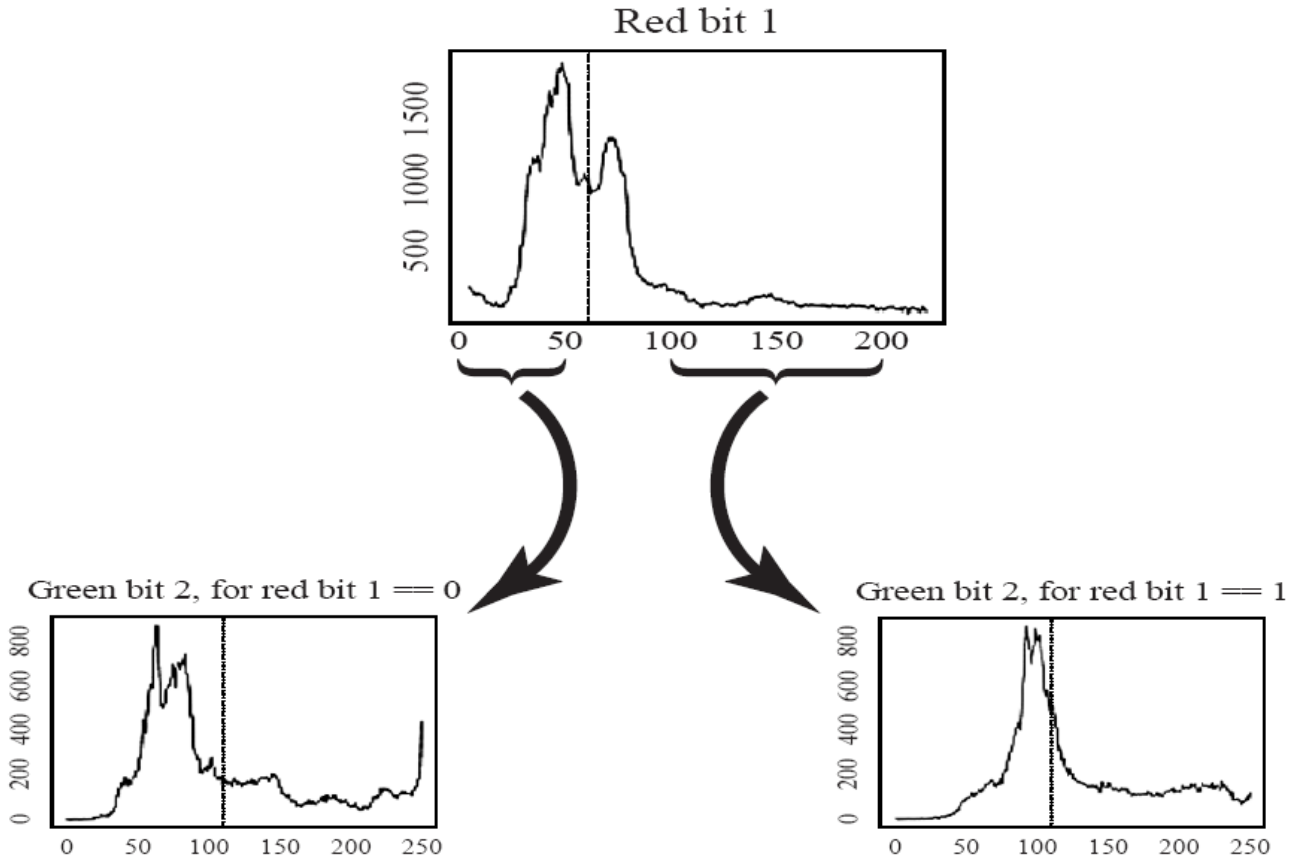

Fig. 3.11: Histogram of R bytes for the 24-bit color image "forestfire.bmp" results in a "0" bit or "1" bit label for every pixel. For the second bit of the color table index being built, we take R values less than the R median and label just those pixels as "0" or "1" according as their G value is less than or greater than the median of the G value, just for the "0" Red bit pixels. Continuing over R, G, B for 8 bits gives a color LUT 8-bit index.

3.2 Popular File Formats

- **8-bit GIF**: one of the most important formats because of its historical connection to the WWW and HTML markup language as the first image type recognized by net browsers.
- **JPEG**: currently the most important common file format.
- **PNG:** most popular lossless image format.
- **TIFF:** flexible file format due to the addition of tags.
- **EXIF:** allows the addition of image metadata.
- **PS and PDF:** vector based language, popular in publishing and academia

3.2.1 GIF

- **GIF standard**: (We examine GIF standard because it is so simple! yet contains many common elements.)

  Limited to 8-bit (256) color images only, which, while producing acceptable color images, is best suited for images with few distinctive colors (e.g., graphics or drawing).

- GIF standard supports **interlacing** — successive display of pixels in widely-spaced rows by a 4-pass display process.

- GIF actually comes in two flavors:

  1. **GIF87a**: The original specification.
  2. **GIF89a**: The later version. Supports simple animation via a Graphics Control Extension block in the data, provides simple control over delay time, a transparency index, etc.

GIF87

- For the standard specification, the general file format of a GIF87 file is as in Fig. 3.12.

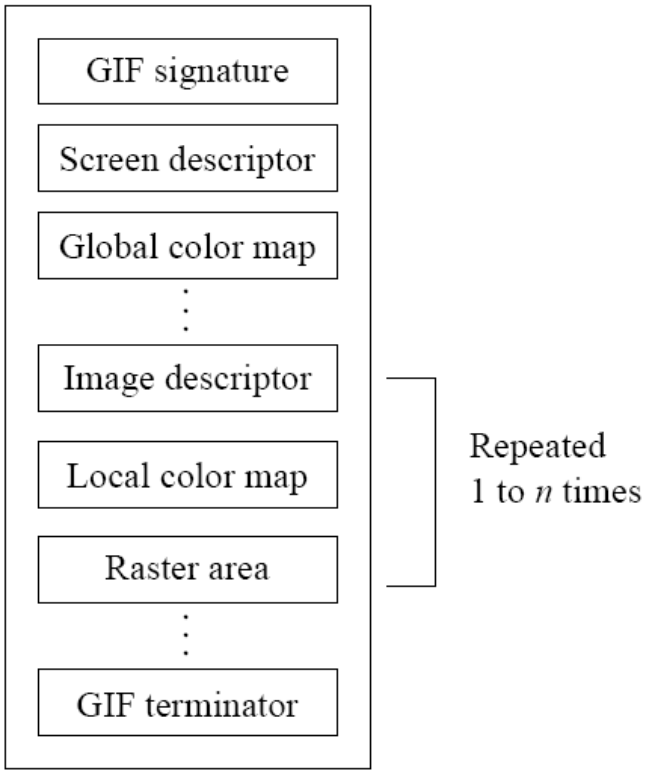

Fig. 3.12: GIF file format.

- **Screen Descriptor** comprises a set of attributes that belong to every image in the file. According to the GIF87 standard, it is defined as in Fig. 3.13.

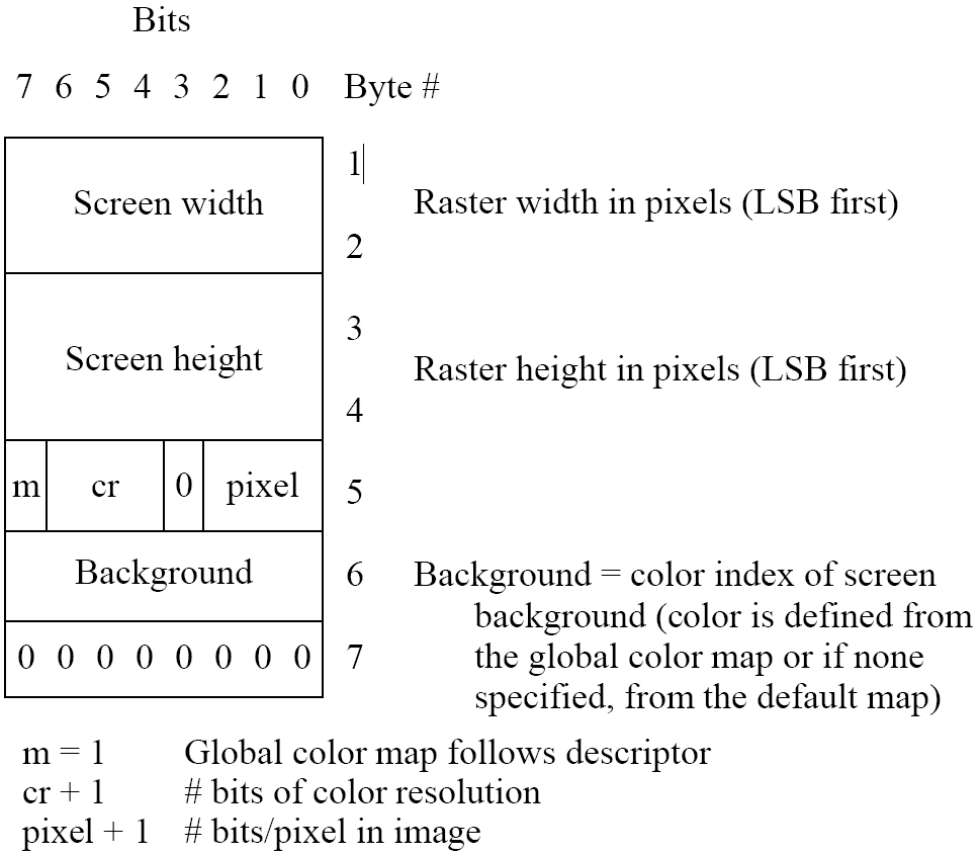

Fig. 3.13: GIF screen descriptor

- **Color Map** is set up in a very simple fashion as in Fig. 3.14. However, the actual length of the table equals $2^{(pixel+1)}$ as given in the Screen Descriptor.

Bits 7 6 5 4 3 2 1 0	Byte #	
Red intensity	1	Red value for color index 0
Green intensity	2	Green value for color index 0
Blue intensity	3	Blue value for color index 0
Red intensity	4	Red value for color index 1
Green intensity	5	Green value for color index 1
Blue intensity	6	Blue value for color index 1
⋮		(continues for remaining colors)

Fig. 3.14: GIF color map.

- Each image in the file has its own **Image Descriptor**, defined as in Fig. 3.15.

Bits 7	6	5	4	3	2	1	0	Byte #	
0	0	1	0	1	1	0	0	1	Image separator character (comma)
Image left								2 3	Start of image in pixels from the left side of the screen (LSB first)
Image top								4 5	Start of image in pixels from the top of the screen (LSB first)
Image width								6 7	Width of the image in pixels (LSB first)
Image height								8 9	Height of the image in pixels (LSB first)
m	i	0	0	0	pixel			10	m = 0 Use global color map, ignore 'pixel' m = 1 Local color map follows, use 'pixel' i = 0 Image formatted in Sequential order i = 1 Image formatted in Interlaced order pixel + 1 # bits per pixel for this image

Fig. 3.15: GIF image descriptor.

- If the "interlace" bit is set in the local Image Descriptor, then the rows of the image are displayed in a four-pass sequence (Fig.3.16).

Image row	Pass 1	Pass 2	Pass 3	Pass 4	Result
0	*1a*				*1a*
1				*4a*	*4a*
2			*3a*		*3a*
3				*4b*	*4b*
4		*2a*			*2a*
5				*4c*	*4c*
6			*3b*		*3b*
7				*4d*	*4d*
8	*1b*				*1b*
9				*4e*	*4e*
10			*3c*		*3c*
11				*4f*	*4f*
12		*2b*			*2b*
⋮					

Fig. 3.16: GIF 4-pass interlace display row order.

- We can investigate how the file header works in practice by having a look at a particular GIF image. Fig. 3.7 on page is an 8-bit color GIF image, in UNIX, issue the command:

```
od -c forestfire.gif | head -2
```

and we see the first 32 bytes interpreted as characters:

```
G  I  F  8  7  a  \208  \2  \188  \1  \247  \0  \0  \6  \3  \5
J  \132  \24  |  )  \7  \198  \195  \  \128  U  \27  \196  \166  &  T
```

- To decipher the remainder of the file header (after “GIF87a”), we use hexadecimal:

```
od -x forestfire.gif | head -2
```

with the result

```
4749 4638 3761 d002 bc01 f700 0006 0305 ae84 187c 2907 c6c3 5c80
551b c4a6 2654
```

3.2.2 JPEG

- **JPEG**: The most important current standard for image compression.

- The human vision system has some specific limitations and JPEG takes advantage of these to achieve high rates of compression.

- JPEG allows the user to set a desired level of quality, or compression ratio (input divided by output).

- As an example, Fig. 3.17 shows our **forestfire** image, with a quality factor Q=10.
  - This image is a mere 1.5% of the original size. In comparison, a JPEG image with Q=75 yields an image size 5.6% of the original, whereas a GIF version of this image compresses down to 23.0% of uncompressed image size.

Fig. 3.17: JPEG image with low quality specified by user.

PNG

- **PNG format**: standing for **Portable Network Graphics** — meant to supersede the GIF standard, and extends it in important ways.

- Special features of PNG files include:

  1. Support for up to 48 bits of color information — a large increase.

  2. Files may contain gamma-correction information for correct display of color images, as well as alpha-channel information for such uses as control of transparency.

  3. The display progressively displays pixels in a 2-dimensional fashion by showing a few pixels at a time over seven passes through each 8 x 8 block of an image.

TIFF

- **TIFF**: stands for **Tagged Image File Format**.
- The support for attachment of additional information (referred to as "tags") provides a great deal of flexibility.
  1. The most important tag is a format signifier: what type of compression etc. is in use in the stored image.
  2. TIFF can store many different types of image: 1-bit, grayscale, 8-bit color, 24-bit RGB, etc.
  3. TIFF was originally a lossless format but now a JPEG tag allows one to opt for JPEG compression.
  4. The TIFF format was developed by the Aldus Corporation in the 1980's and was later supported by Microsoft.

EXIF

- **EXIF** (Exchange Image File) is an image format for digital cameras:

1. Compressed EXIF files use the baseline JPEG format.

2. A variety of tags (many more than in TIFF) are available to facilitate higher quality printing, since information about the camera and picture-taking conditions (flash, exposure, light source, white balance, type of scene, etc.) can be stored and used by printers for possible color correction algorithms.

3. The EXIF standard also includes specification of file format for audio that accompanies digital images. As well, it also supports tags for information needed for conversion to FlashPix (initially developed by Kodak).

PS and PDF

- **Postscript** is an important language for typesetting, and many high-end printers have a Postscript interpreter built into them.

- Postscript is a vector-based picture language, rather than pixel-based: page element definitions are essentially in terms of vectors.

  1. Postscript includes text as well as vector/structured graphics.

  2. Bit-mapped images can be included in output files.

  3. Encapsulated Postscript files (.EPS) add some additional information for inclusion of Postscript files in another document.

4. Postscript page description language itself does not provide compression; in fact, Postscript files are just stored as ASCII.

- Another text + figures language has superseded or at least paralleled Postscript: Adobe Systems Inc. includes LZW compression in its Portable Document Format (**PDF**) file format.
  - PDF files that do not include images have about the same compression ratio, 2:1 or 3:1, as do files compressed with other LZW-based compression tools.

Some Other Image Formats

- **Microsoft Windows: WMF**: the native vector file format for the Microsoft Windows operating environment:
  1. Consist of a collection of GDI (Graphics Device Interface) function calls, also native to the Windows environment.
  2. When a WMF file is "played" (typically using the Windows PlayMetaFile() function) the described graphics is rendered.
  3. WMF files are ostensibly device-independent and are unlimited in size.

- **Microsoft Windows: BMP**: the major system standard graphics file format for Microsoft Windows, recognized by many programs. Watch it!: there are many sub-variants within the BMP standard.

- **Netpbm Format**: PPM (Portable PixMap), PGM (Portable GrayMap), and PBM (Portable BitMap) belong to a family of open source Netpbm formats. These formats are mostly common in the linux/unix environments.

PTM

- PTM (Polynomial Texture Mapping) is a technique for storing a representation of a camera scene that contains information about a set of images taken under a set of lights that each have the same spectrum (say, a xenon flash), but with each light placed at a different direction from the scene. PTM was invented at Hewlett-Packard.

Fig. 3.18: (a) 50 input images for PTM: lights individually from 50 different directions $\boldsymbol{e}_i$, i=1..50; (b) interpolated image under *new* light $\boldsymbol{e}$.

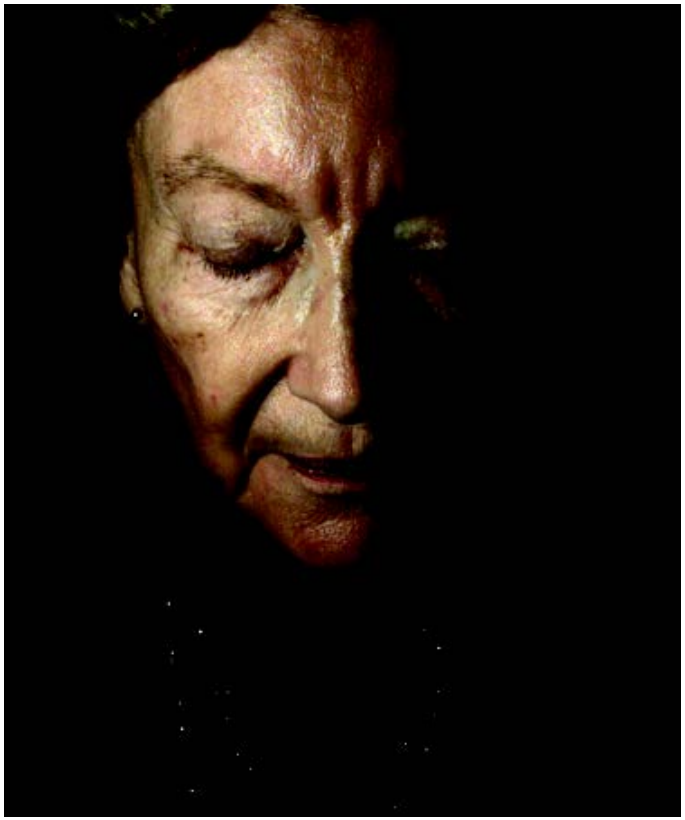

Fig. 3.18:

(a) 50 input images for PTM: lights individually from 50 different directions $\boldsymbol{e}_i$, i=1..50;

(b) interpolated image under *new* light ***e***