Chapter 3
Image Data Representations

Outline

- Image data types:
  - Binary image (1-bit image)
  - Gray-level image (8-bit image)
  - Color image (8-bit and 24-bit images)
- Image file formats:
  - GIF, JPEG, PNG, TIFF, EXIF

Graphics and 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 Macromedia Director.

Table 3.1: Macromedia Director File Formats

File Import					File Export		Native
Image	Palette	Sound	Video	Anim.	Image	Video	
.BMP, .DIB, .GIF, .JPG, .PICT, .PNG, .PNT, .PSD, .TGA, .TIFF, .WMF	.PAL .ACT	.AIFF .AU .MP3 .WAV	.AVI .MOV	.DIR .FLA .FLC .FLI .GIF .PPT	.BMP	.AVI .MOV	.DIR .DXR .EXE

1-bit Image - binary image

- 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. It is satisfactory for pictures containing only simple graphics and text.
- A 640X480 monochrome image requires 38.4 kilobytes of storage (= 640x480 / (8x1000)).

8-bit Gray-level Images

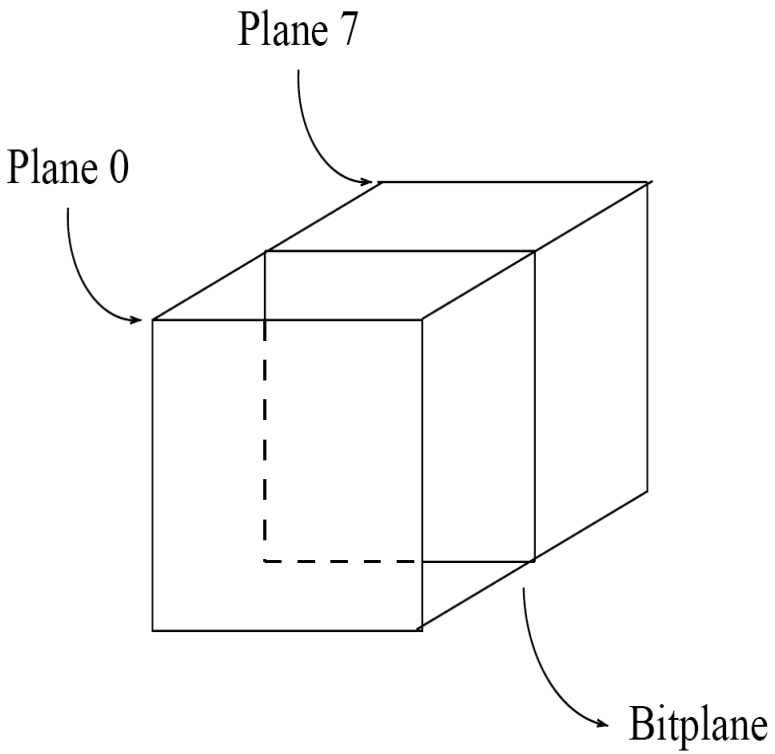
- Each pixel has a gray-value between 0 and 255. Each pixel is represented by a single byte; e.g., a dark pixel might have a value of 10, and a bright one might be 230.

- **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 1,600 x1,200, whereas lower resolution might be 640 x 480. As noticed numbers are in as aspect ratio 4:3 because it is found that image looks natural.

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

- 8-bit image can be thought of as a set of 1-bit **bit-planes**, where each plane consists of a 1-bit representation of the image at higher and higher levels of "elevation": a bit is turned on if the image pixel has a nonzero value that is at or above that bit level.
- Each bit-plane can have a value of 0 or 1 at each pixel but, together, all the bit-planes make up a single byte that stores values between 0 and 255 (in this 8-bit situation)
- 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).



Bit-planes for 8-bit grayscale image.

Grayscale image of Lena.

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**.

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, various forms of distortion, artistic effects, geometric transforms and texture effects).

- 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.

(a)

(b)

(c)

(d)

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

8-bit Color Images

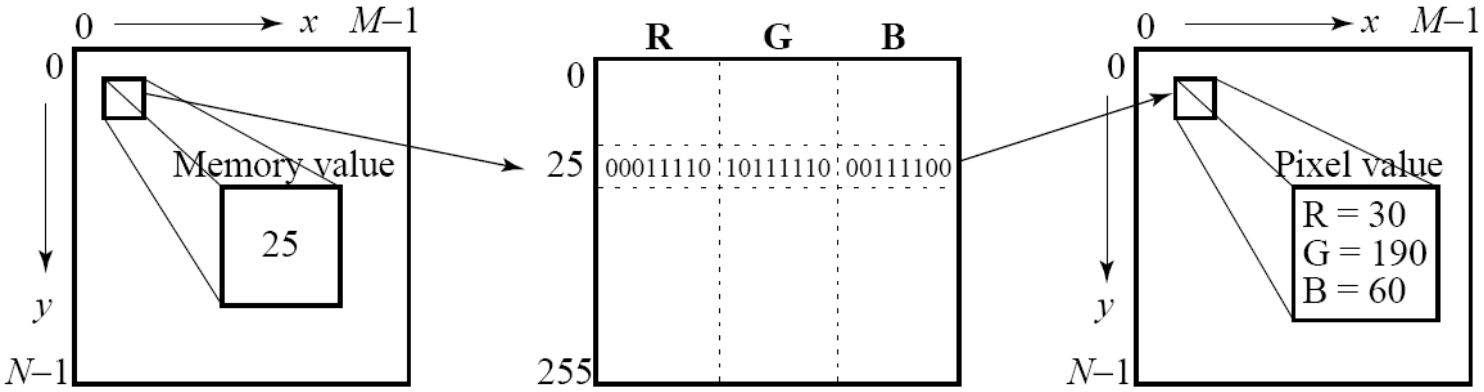
- When space is a concern, reasonably accurate color images can be obtained by quantizing the color information to collapse it.
- Systems can make use of 8-bits of color information (the so-called “256 colors”) in producing a screen image.
- Such image files use the concept of a **lookup table** to store 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.

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).

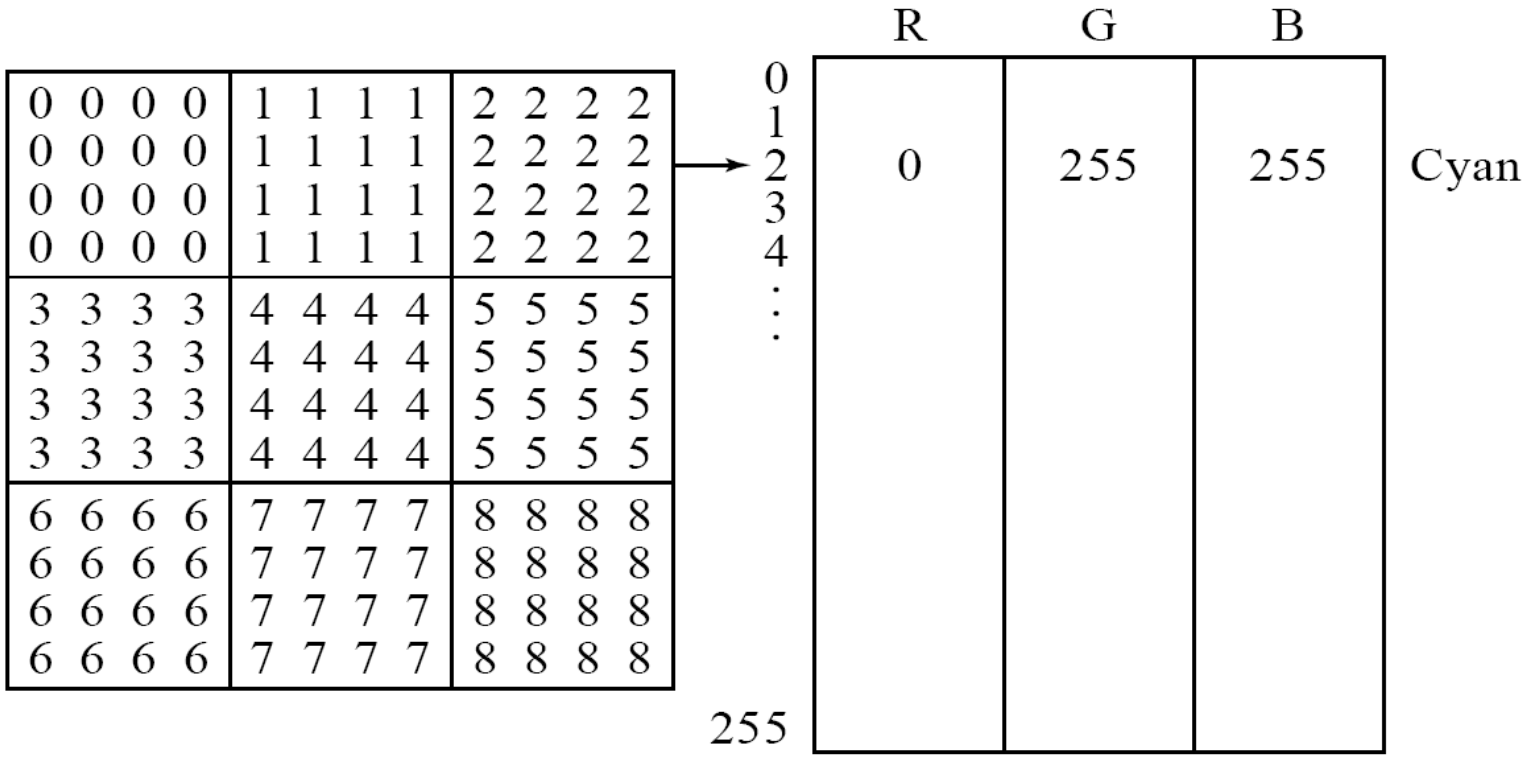
Color Look-up Tables (LUTs)

- 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).
- Images stored in row-column order and displayed as two-dimensional array.



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. in next slide 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**.



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

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.
  - 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.
  - 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.
  - 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.
- However, what tends to happen with this simple scheme is that edge artifacts appear in the image. The reason is that if a slight change in RGB results in shifting to a new code, an edge appears, and this can quite annoying perceptually.

- **Median-cut algorithm**: A simple alternate solution that does a better job for this color reduction problem.
- The method is a type of adaptive partitioning scheme that tries to put the most bits, the most discrimination power, where colors are most clustered

  (a) The idea is to sort the R byte values and find their median; then values smaller than the median are labeled with a “0” bit and values larger than the median are labeled 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.

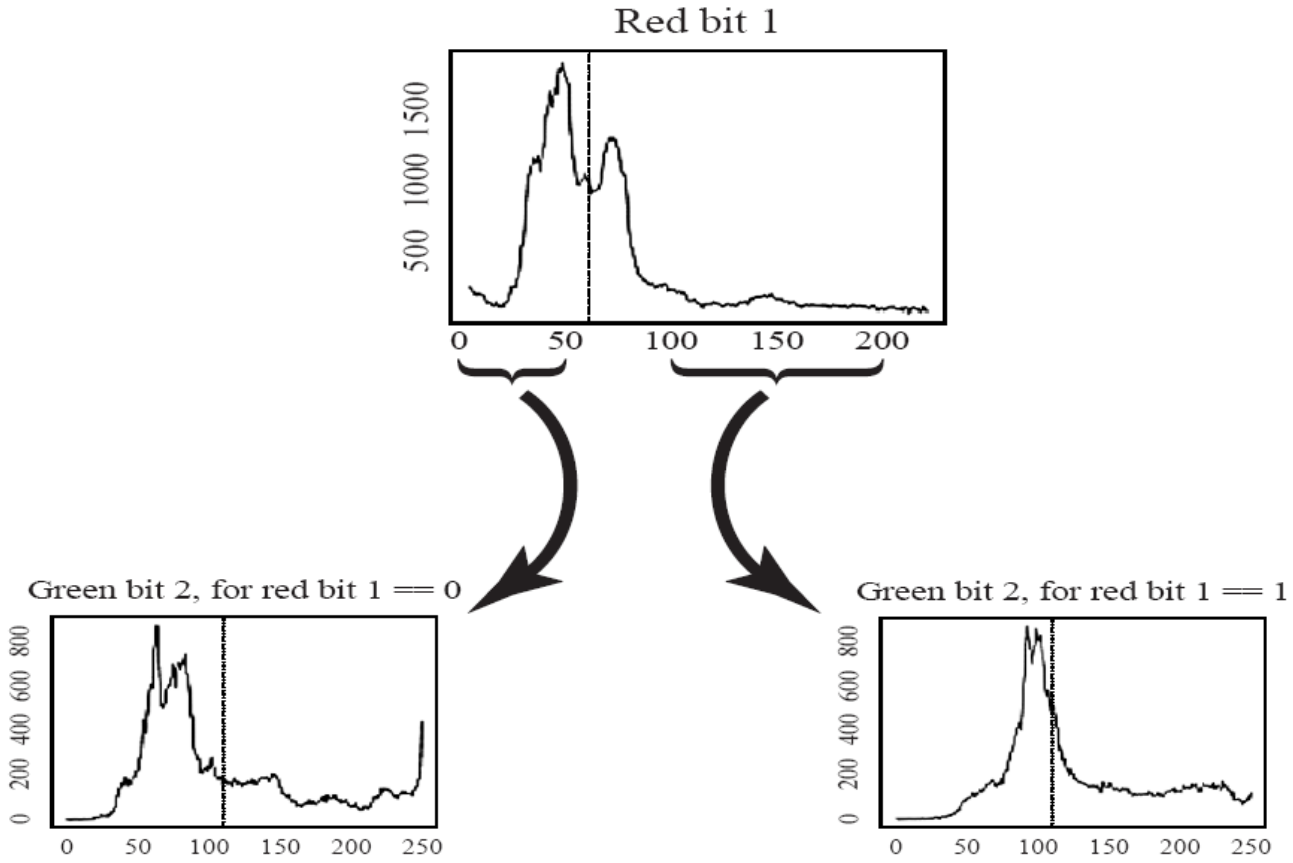


Fig. 3.11 Histogram of R bytes for the 24-bit color image "forestfire.bmp" results in a "0" bit or "1" bit label for every pixel. For the second bit of the color table index being built, we take R values less than the R median and label just those pixels as "0" or "1" according as their G value is less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