CMPT 365 Multimedia Systems

Media Representations - Image

Spring 2017

Edited from slides by Dr. Jiangchuan Liu

Outline

- Color Science
- Color space: RGB
- Color Space: YUV and more
- Graphics/Image Data Types
  - Popular File Formats

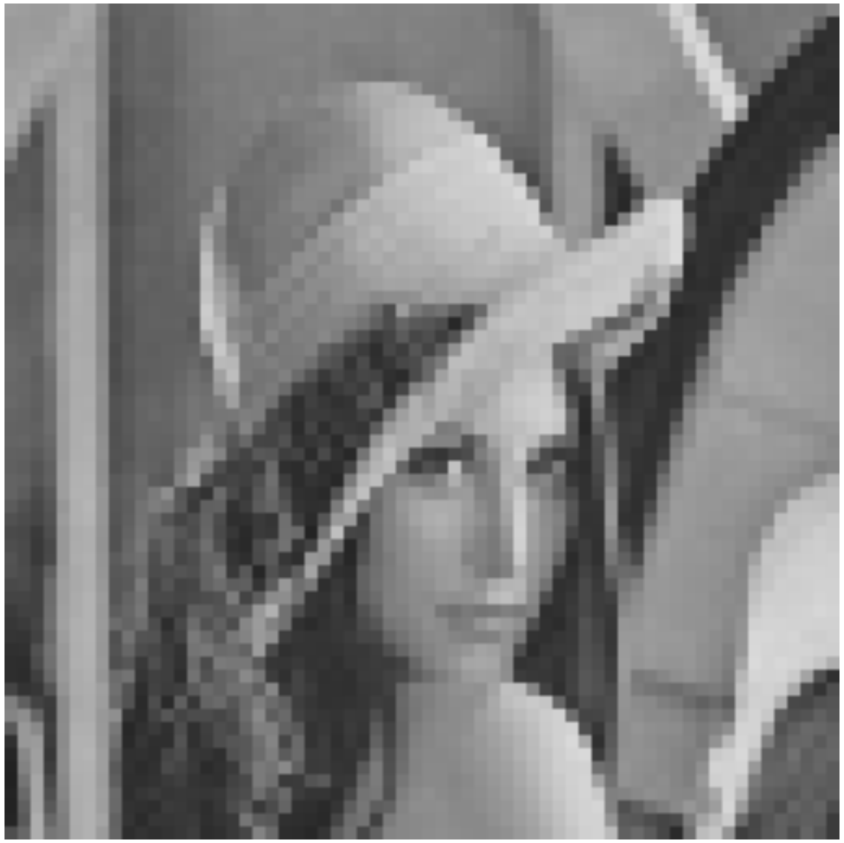
What is image/picture ?

- A 2D signal
  - X*Y pixels after sampling
  - Captured by CCD/CMOS
- Each pixel has a color

Image (2D)->

Pixel ->

Color



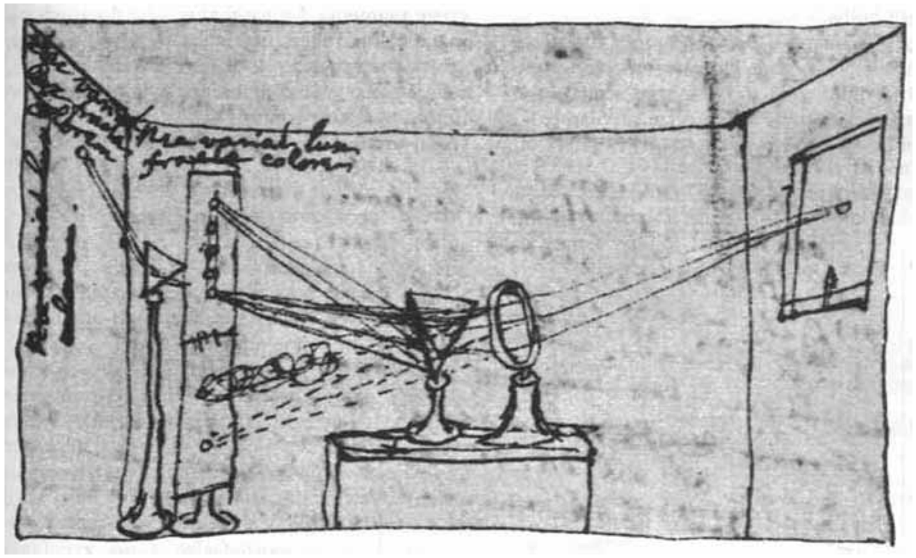
- color itself is a multi-dimensional vector, unless for black/white (0/1) or grayscale (scalar)

Color Science

- Light and Spectra
  - Light is an electromagnetic wave. Its color is characterized by the wavelength content of the light
    - Short wavelengths produce a blue sensation, long wavelengths produce a red one
  - Laser light consists of a single wavelength: e.g., a ruby laser produces a bright, scarlet-red beam
  - Most light sources produce contributions over many wavelengths
  - However, humans cannot detect all light, just contributions that fall in the “visible wavelengths"

Color Science

- Visible light
  - Electromagnetic wave in the range 400 nm to 700 nm (where nm stands for nanometer, 10–9 meters)



White light contains all the colors of a rainbow.

Sir Isaac Newton's experiments. *By permission of the Warden and Fellows, New College, Oxford.*

- Fig. 4.2 shows the relative power in each wavelength interval for typical outdoor light on a sunny day. This type of curve is called a Spectral Power Distribution (SPD) or a spectrum.
- The symbol for wavelength is λ. This curve is called $E(\lambda)$.

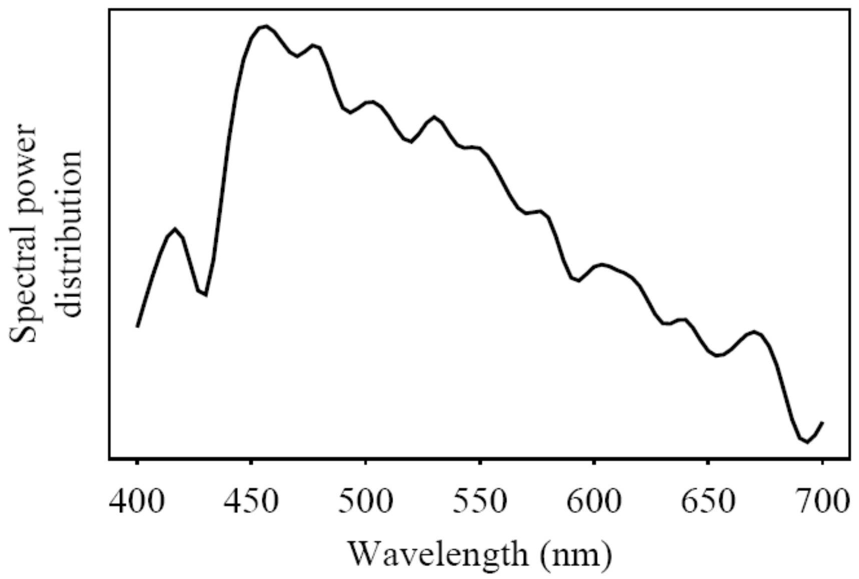
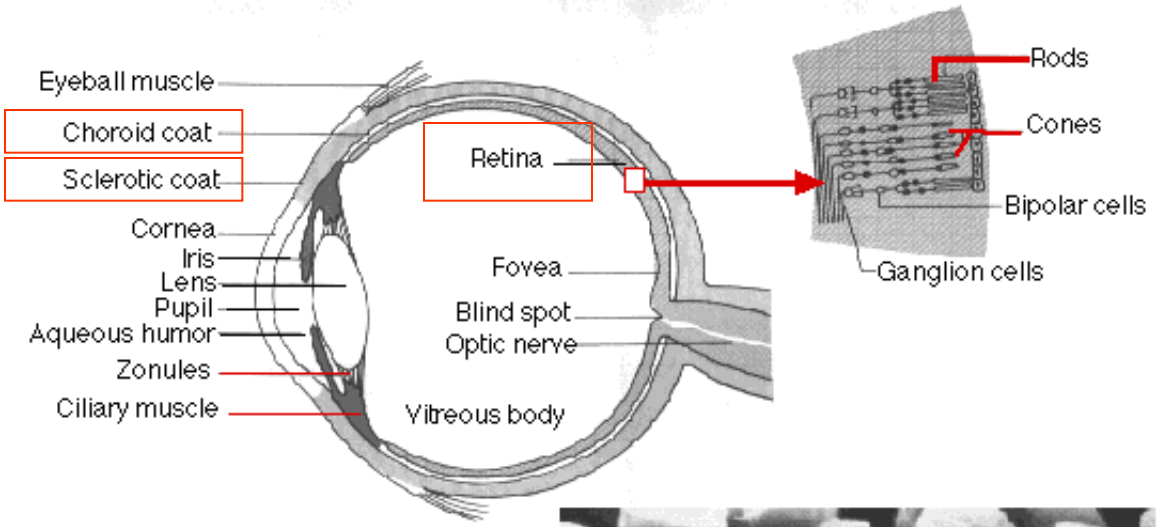
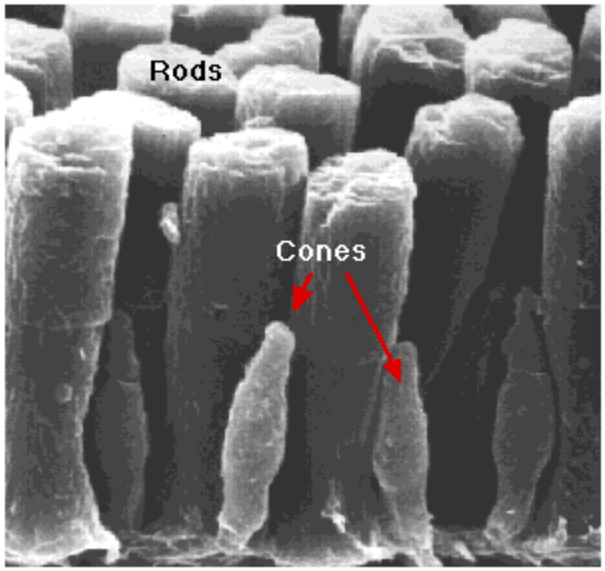


Fig. 4.2: Spectral power distribution of daylight.

Human Visual System





- Three layers:
  - Sclerotic coat
    - Includes cornea
  - Choroid coat
    - Includes iris, pupil
  - Retina: contains light receptors
    - Rods
    - Cones that absorb red light (long-wavelength)
    - Cones that absorb green light
    - Cones that absorb blue light (short wl)

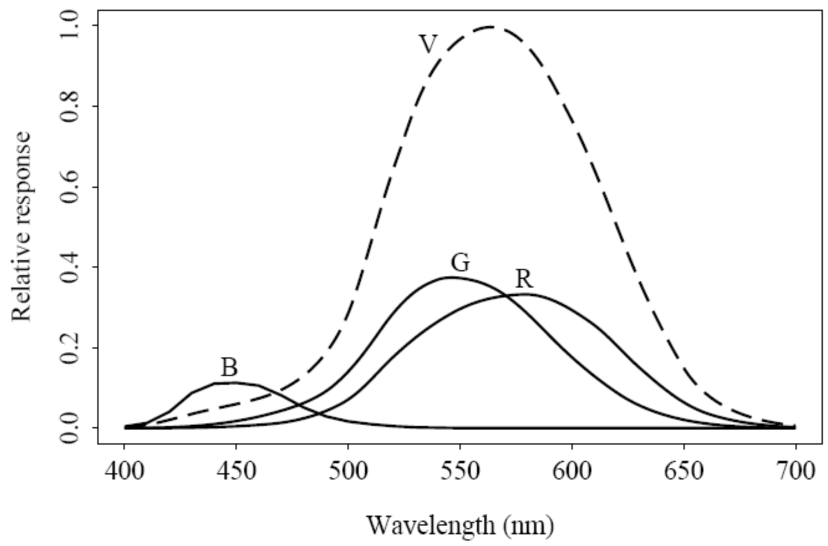
http://users.rcn.com/jkimball.ma.ultranet/BiologyPages/V/Vision.html

4.1.3 Spectral Sensitivity of the Eye

- The eye is most sensitive to light in the middle of the visible spectrum.
- The sensitivity of our receptors is also a function of wavelength (Fig. 4.3 below).
- The Blue receptor sensitivity is not shown to scale because it is much smaller than the curves for Red or Green — Blue is a late addition, in evolution.
  - Statistically, Blue is the favourite color of humans, regardless of nationality — perhaps for this reason: Blue is a latecomer and thus is a bit surprising!
- Fig. 4.3 shows the overall sensitivity as a dashed line — this important curve is called the luminous-efficiency function.
  - It is usually denoted $V(\lambda)$ and is formed as the sum of the response curves for Red, Green, and Blue.

- The rod sensitivity curve looks like the luminous-efficiency function $V(\lambda)$ but is shifted to the red end of the spectrum.

- The achromatic channel produced by the cones is approximately proportional to 2R+G+B/20.



❒ **Fig. 4.3**: R,G, and B cones, and Luminous Efficiency curve $V(\lambda)$.

- These spectral sensitivity functions are usually denoted by letters other than “R,G,B”; here let’s use a vector function $q(\lambda)$, with components

$$q(\lambda) = (q_R(\lambda), q_G(\lambda), q_B(\lambda))^T \tag{4.1}$$

- The response in each color channel in the eye is proportional to the number of neurons firing.

- A laser light at wavelength λ would result in a certain number of neurons firing. An SPD is a combination of single-frequency lights (like “lasers”), so we add up the cone responses for all wavelengths, weighted by the eye’s relative response at that wavelength.

- We can succinctly write down this idea in the form of an integral:

$$R = \int E(\lambda)\, q_R(\lambda)\, d\lambda$$

$$G = \int E(\lambda)\, q_G(\lambda)\, d\lambda$$

$$B = \int E(\lambda)\, q_B(\lambda)\, d\lambda \qquad (4.2)$$

4.1.4 Image Formation

- Surfaces reflect different amounts of light at different wavelengths, and dark surfaces reflect less energy than light surfaces.

- Fig. 4.4 shows the surface spectral reflectance from (1) orange sneakers and (2) faded blue jeans. The reflectance function is denoted $S(\lambda)$.

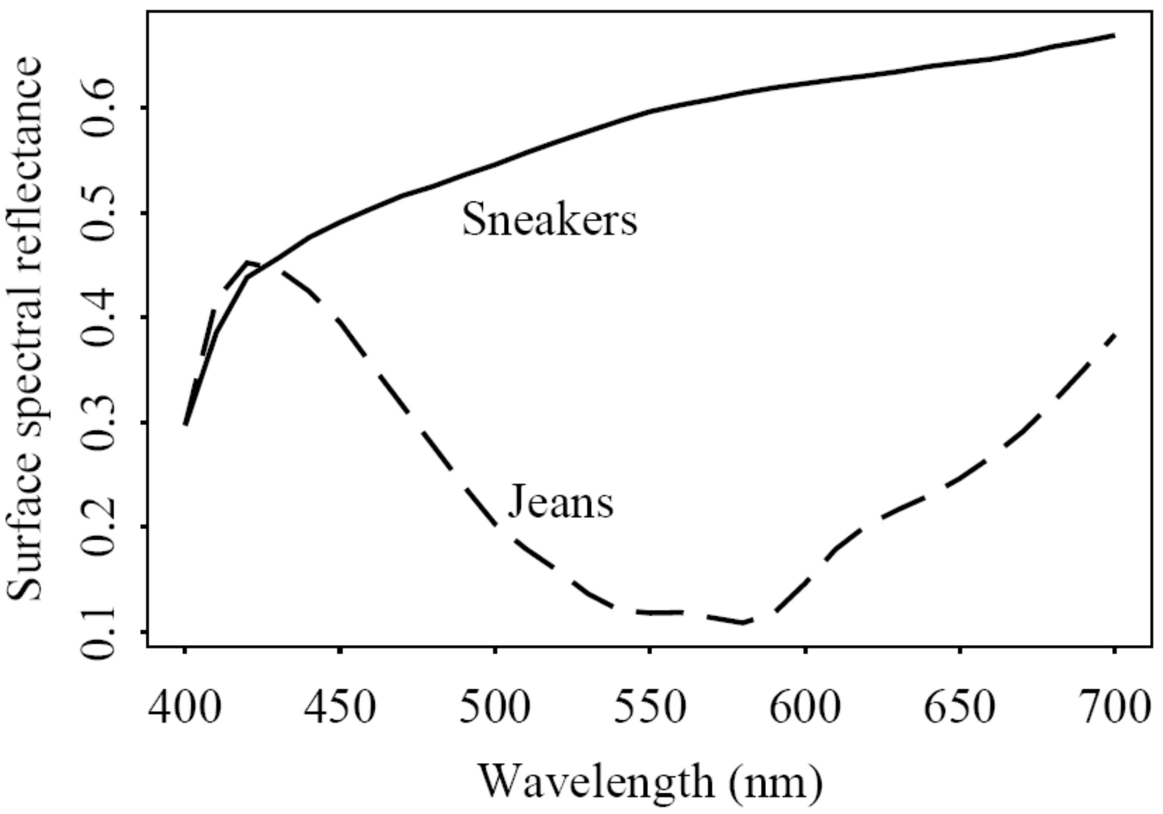


Fig. 4.4: Surface spectral reflectance functions $S(\lambda)$ for objects.

- Image formation is thus:
  - Light from the illuminant with SPD $E(\lambda)$ impinges on a surface, with surface spectral reflectance function $S(\lambda)$, is reflected, and then is filtered by the eye's cone functions $q(\lambda)$.
  - Reflection is shown in Fig. 4.5 below.
  - The function $C(\lambda)$ is called the color signal and consists of the product of $E(\lambda)$, the illuminant, times $S(\lambda)$, the reflectance:

$$C(\lambda) = E(\lambda)\ S(\lambda).$$

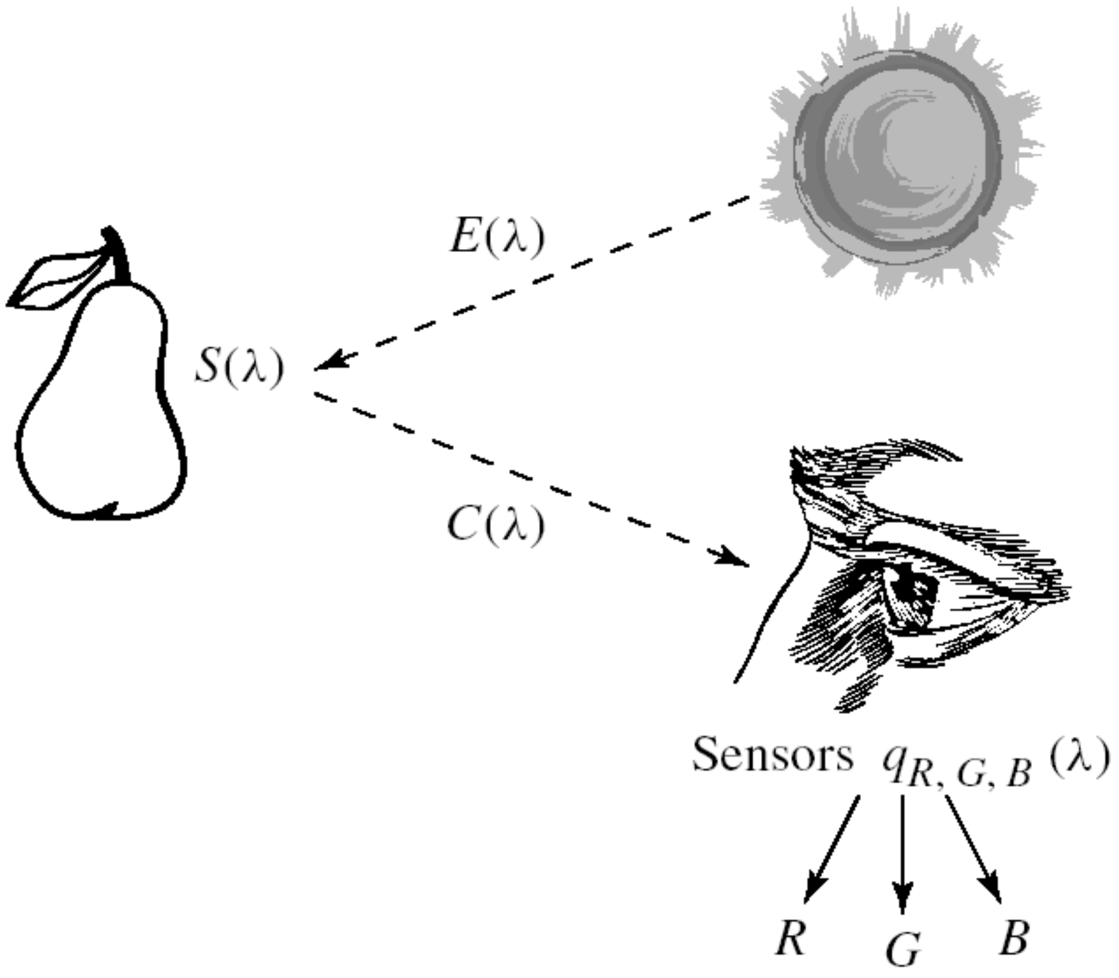


Fig. 4.5: Image formation model.

- The equations that take into account the image formation model are:

$$R = \int E(\lambda)\, S(\lambda)\, q_R(\lambda)\, d\lambda$$

$$G = \int E(\lambda)\, S(\lambda)\, q_G(\lambda)\, d\lambda$$

$$B = \int E(\lambda)\, S(\lambda)\, q_B(\lambda)\, d\lambda \qquad (4.3)$$

4.1.5 Camera Systems

- Camera systems are made in a similar fashion; a studio quality camera has three signals produced at each pixel location (corresponding to a retinal position).

- Analog signals are converted to digital, truncated to integers, and stored. If the precision used is 8-bit, then the maximum value for any of R,G,B is 255, and the minimum is 0.

- However, the light entering the eye of the computer user is that which is emitted by the screen—the screen is essentially a self-luminous source. Therefore we need to know the light $E(\lambda)$ entering the eye.

4.1.6 Gamma Correction

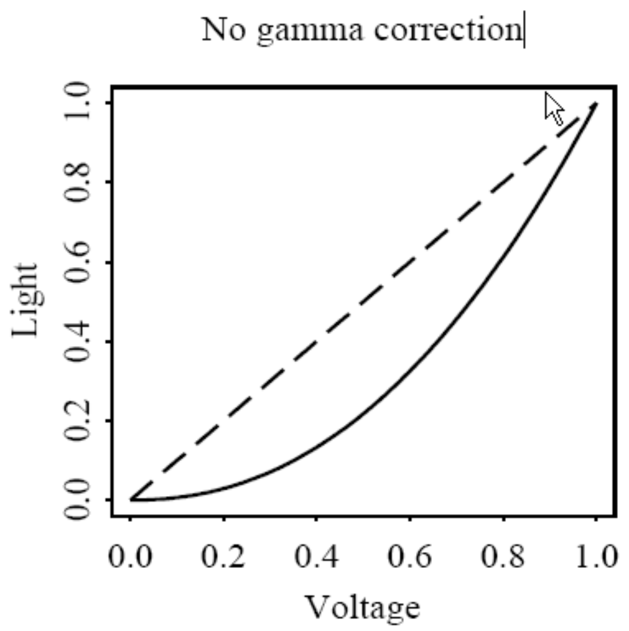
- The light emitted is in fact roughly proportional to the voltage *raised to a power*; this power is called **gamma**, with symbol γ.

(a) Thus, if the file value in the red channel is *R*, the screen emits light proportional to R^{γ}, with SPD equal to that of the red phosphor paint on the screen that is the target of the red channel electron gun. The value of gamma is around 2.2.

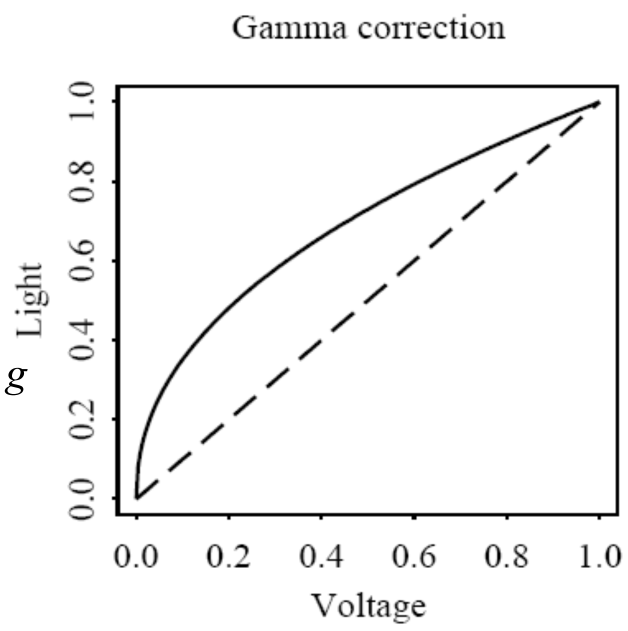
(b) It is customary to append a prime to signals that are **gamma-corrected** by raising to the power $(1/\gamma)$ before transmission. Thus we arrive at **linear signals**:

$$R \to R' = R^{1/\gamma} \Rightarrow (R')^{\gamma} \to R$$

Gamma Correction cont'd

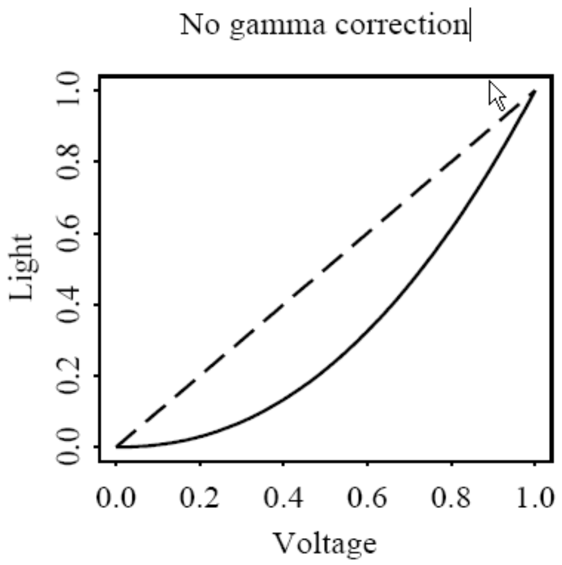


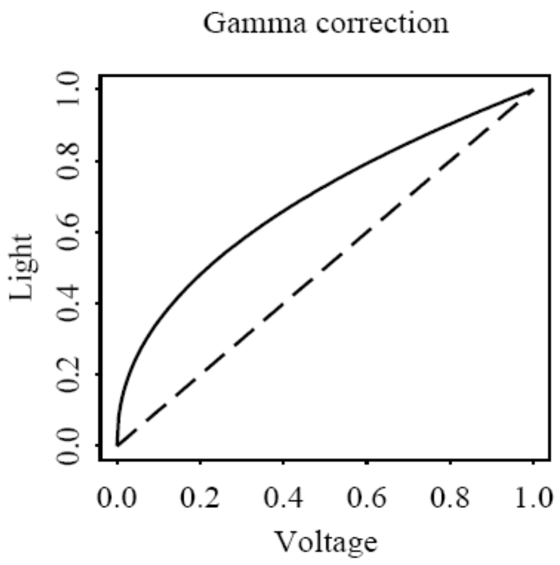
R^{g}

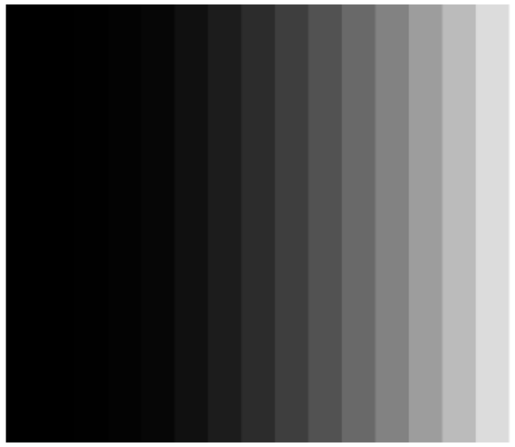


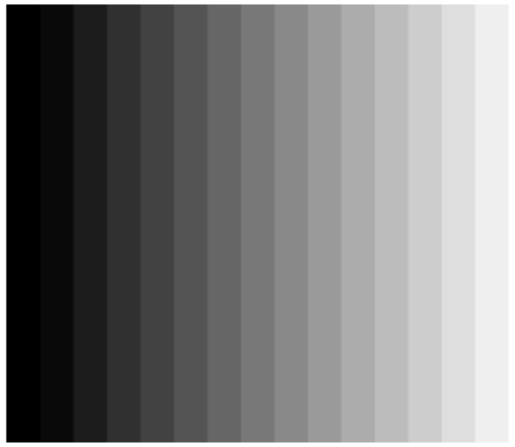
- Left: light output from CRT with no gamma-correction applied. -- Darker values are displayed too dark.
- Right: pre-correcting signals by applying the power law $R^{1/g}$
- Normalization (0-1) ?

Gamma Correction cont'd









- A more careful definition of gamma recognizes that a simple power law would result in an infinite derivative at zero voltage — makes constructing a circuit to accomplish gamma correction difficult to devise in analog.

- In practice a more general transform, such as

❐ $R \rightarrow R' = a \times R^{1/\gamma} + b$ is used, along with special care at the origin:

$$V_{\text{out}} = \begin{cases} 4.5 \times V_{\text{in}}\,, & V_{\text{in}} < 0.018 \\ 1.099 \times (V_{\text{in}}^{0.45} - 0.099)\,, & V_{\text{in}} \geq 0.018 \end{cases}$$ ❐ (4.5)

Correction--Camera

- A 2D signal
  - X*Y pixels after sampling
  - Captured by CCD/CMOS
- Each pixel has a color

Image (2D)->

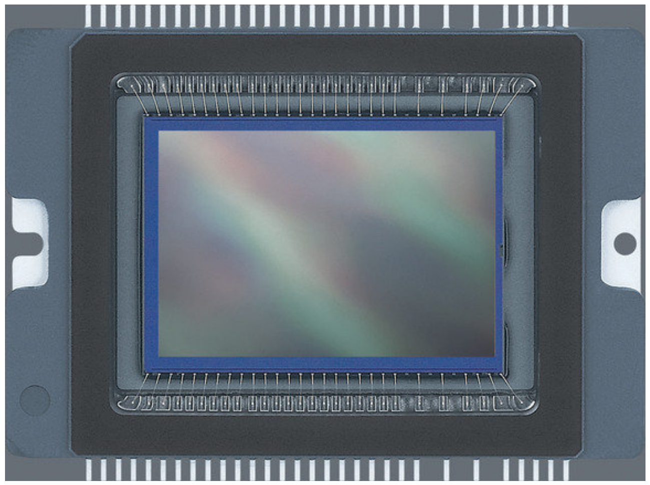
Pixel ->

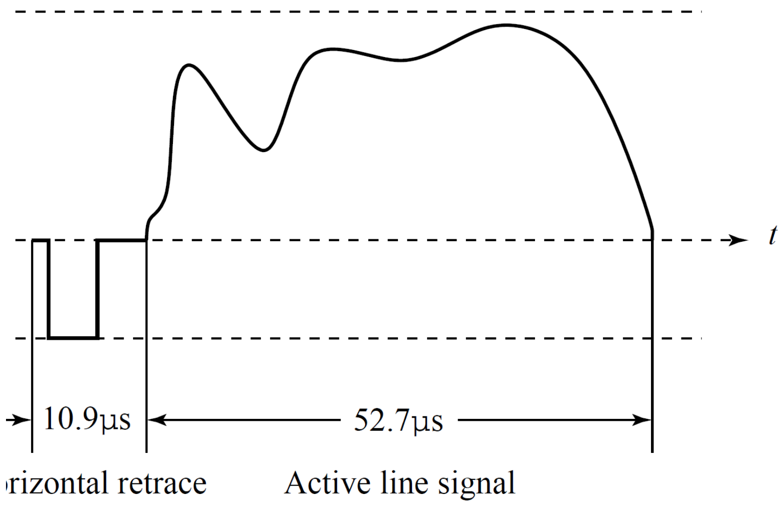
Color

- color itself is a multi-dimensional vector, unless for black/white (0/1) or grayscale (scalar)

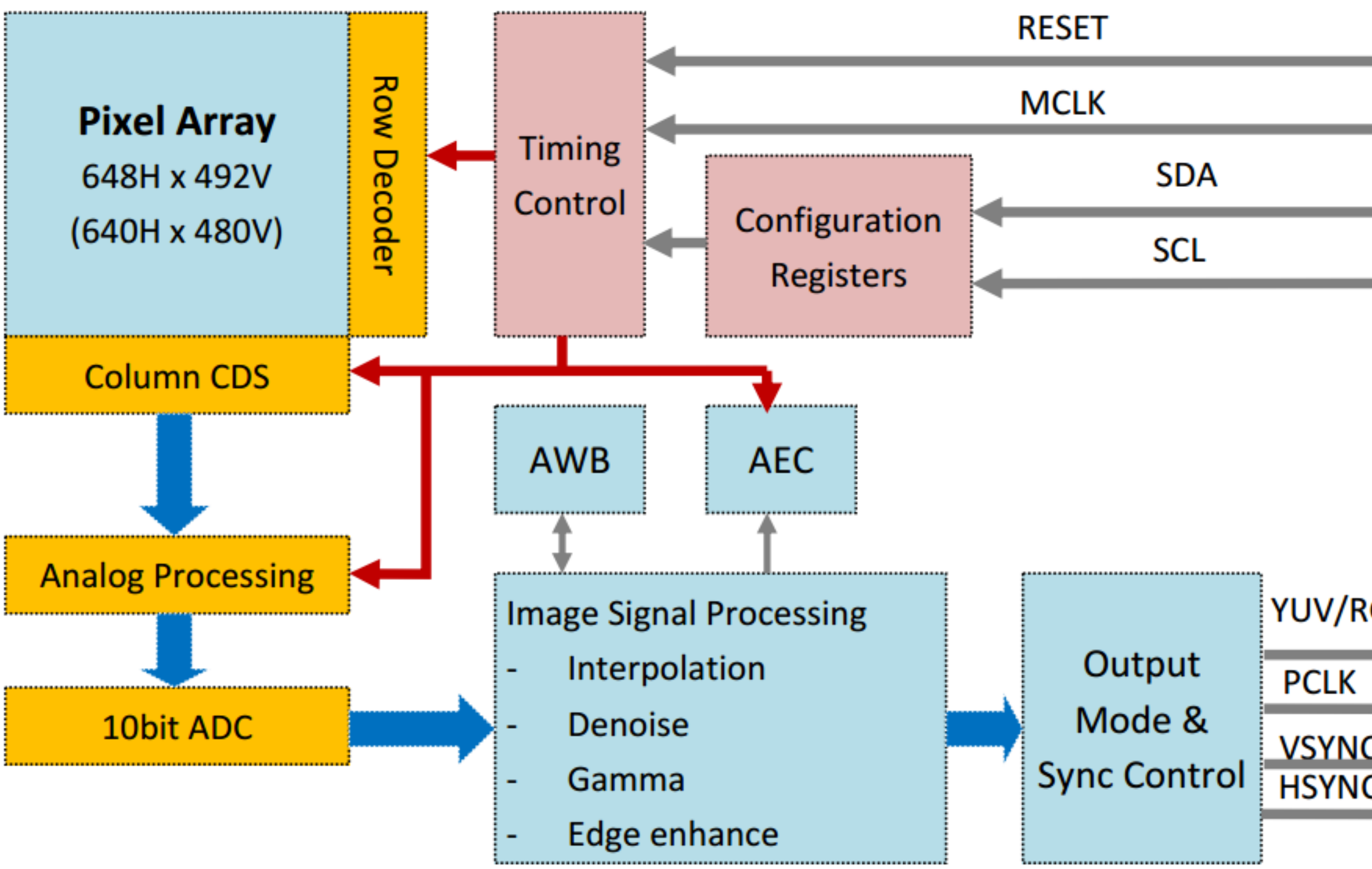
Correction--Camera

- CCD/CMOS are just sensor type
- Usually 2d matrix array
- Scan through each point to produce electronical signal
- Digital camera has Analog to Digital conversion





Correction--Camera

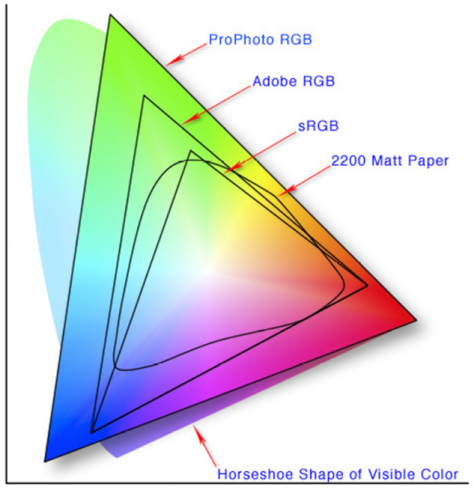


Outline

- Color Science
- Color space: RGB
- Color Space: YUV and more
- Graphics/Image Data Types
    - Popular File Formats

Color Space

- All color can be created by mixing basic components
- Different ways of choosing basic color components
- 1. RGB Color Space
  - R, G, B components
  - Usually 8 bits per components: [0, …, 255]
  - Widely used: BMP, TIFF, PPM …
- 2. RGBA Color Space
  - RGB with Alpha (for transparency)
  - Used by PNG format



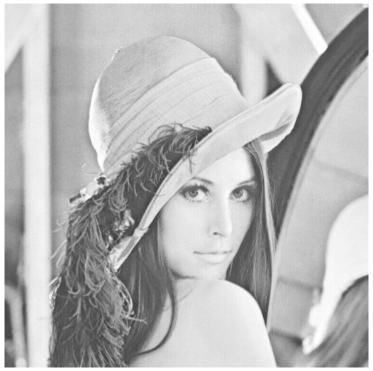
http://en.wikipedia.org/wiki/Color_space

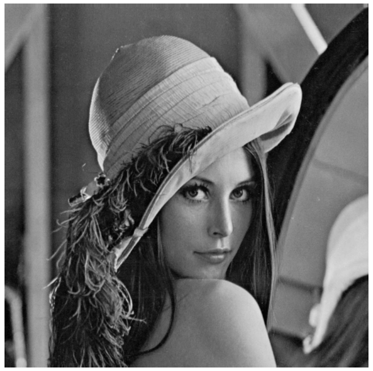
Outline

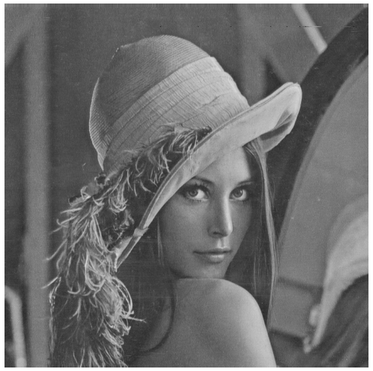
- Color Science
- Color space: RGB
- Color Space: YUV and more
- Graphics/Image Data Types
  - Popular File Formats

Color Space

- RGB components of an image are strongly correlated


R


G


B

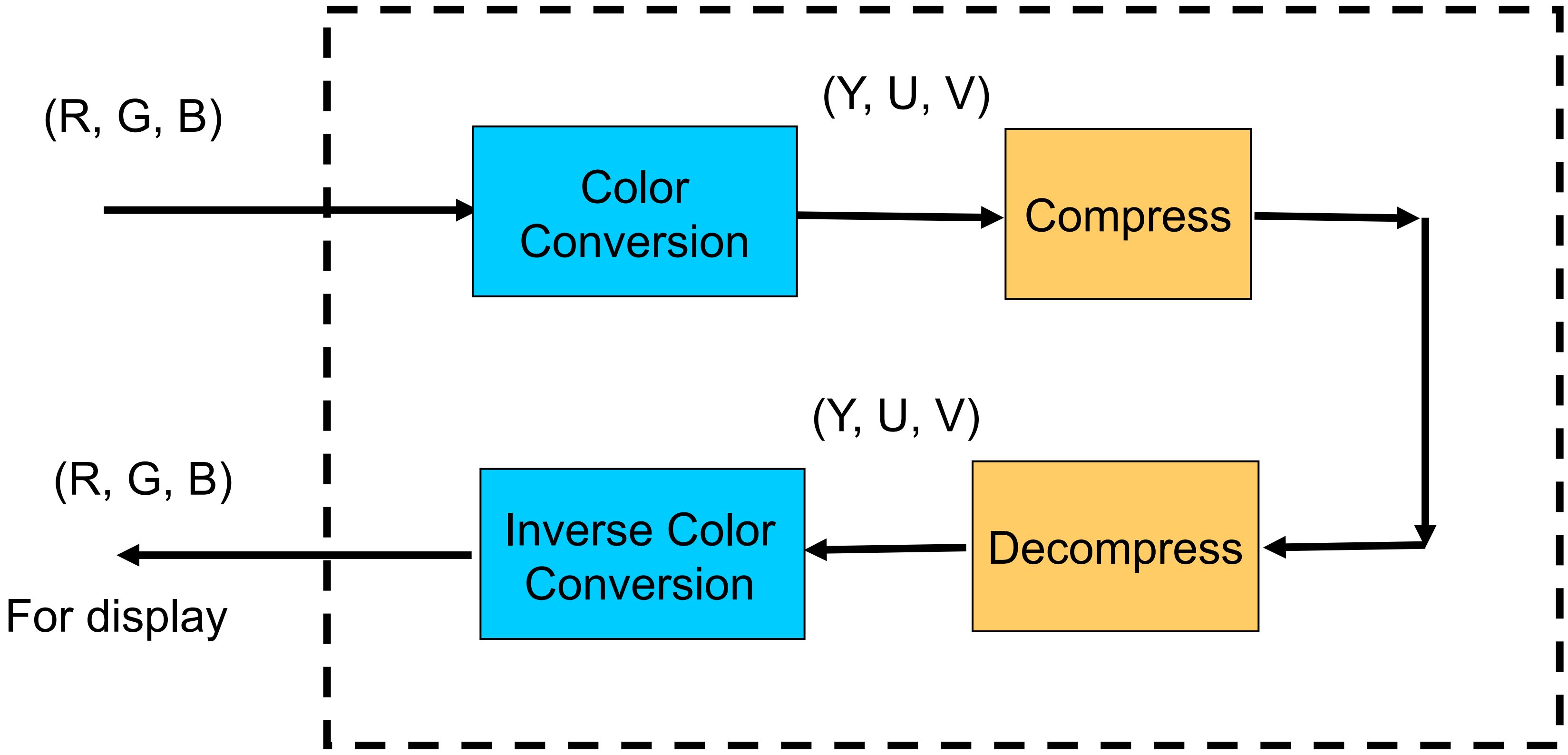
Matlab code:

```
x = imread('lena.ppm');
figure; imshow(x);
figure; imshow(x(:,:,1));
figure; imshow(x(:,:,2));
figure; imshow(x(:,:,3));

Note:
Size of x: (512, 512, 3)
```

Color Space: RGB→YUV

- Solution: convert to other spaces
- Why ? Display device, compression …



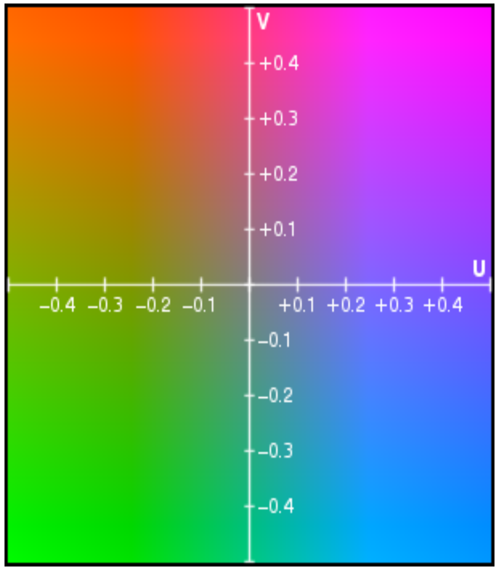
Color Space

- 3. YUV color space (used by PAL TV system)
  - Y: Luminance component (brightness)
  - U, V: Chrominance components

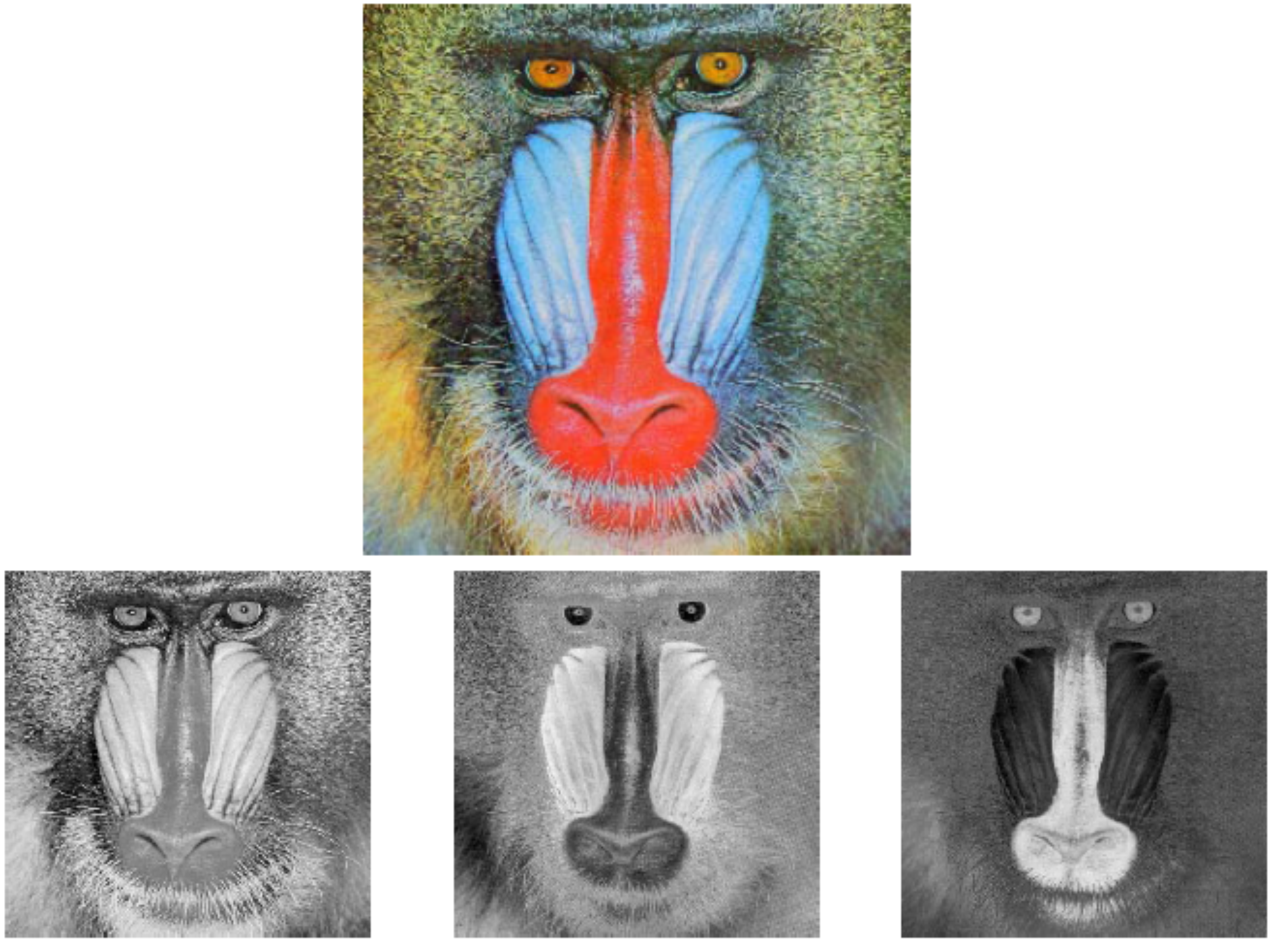
    - the difference between a color and a reference

- 4. **YCrCb Space (used in image/video coding)**
  - Derived from YUV
    - U,V shifted by 0.5
  - Components approximately uncorrelated

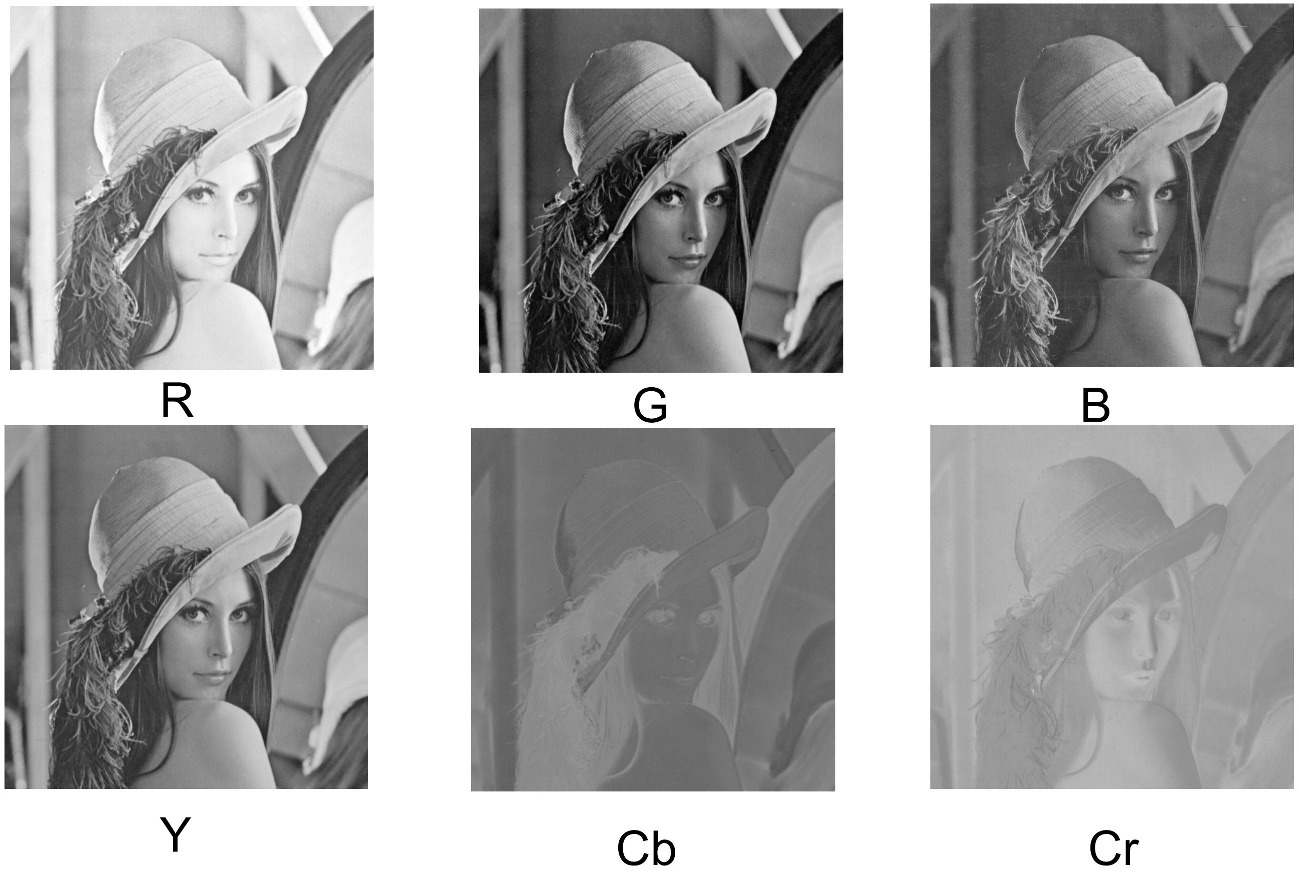
U-V color plane when Y=0.5



YUV Decomposition



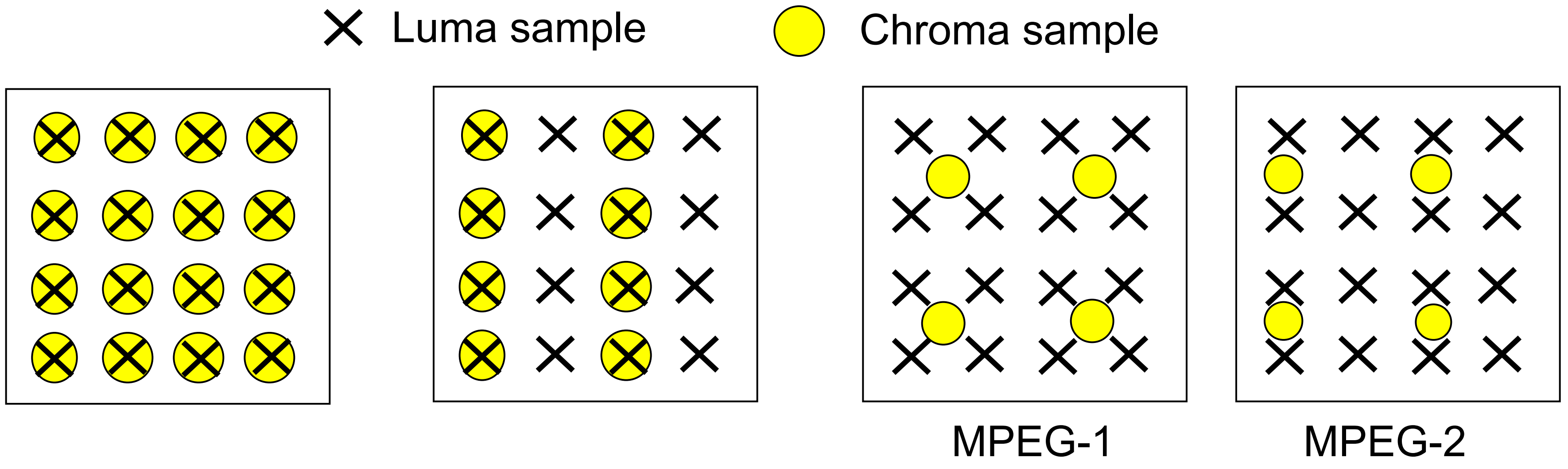
Color Space



- Most information is in Y channel (brightness)
  - Cb and Cr are small → easier for compression
- Human eyes are not sensitive to color error
  - Don't need high resolution for color component

Color Space: Down-sampling

- Down-sampling color components to improve compression



YUV 4:4:4
No downsampling
Of Chroma

YUV 4:2:2
- 2:1 horizontal downsampling of chroma components
- 2 chroma samples for every 4 luma samples

YUV 4:2:0
- 2:1 horizontal downsampling of chroma components
- 1 chroma sample for every 4 luma samples
- Widely used

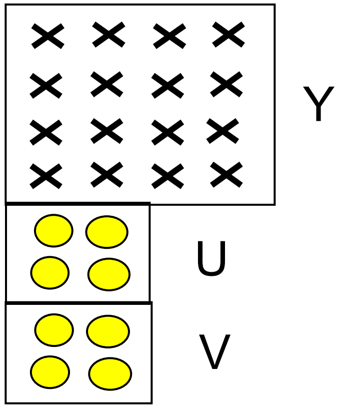
Raw YUV Data File Format

- In YUV 4:2:0, number of U and V samples are 1/4 of the Y samples
- YUV samples are stored separately:

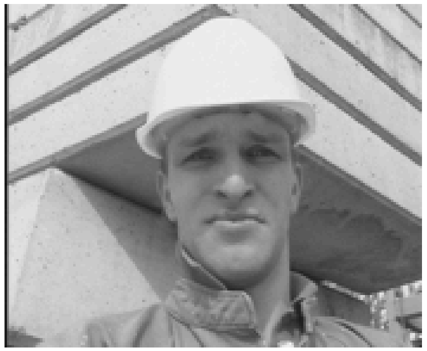
  Image: YYYY…..Y UU…U VV…V

  (row by row in each channel)

  Video: YUV of frame 1, YUV of frame 2, ……



- CIF (Common Intermediate format):
  - 352 x 288 pixels for Y, 176 x 144 pixels for U, V
- QCIF (Quarter CIF): 176 x 144 pixels for Y, 88 x 72 pixels for U, V
- CIF, and QCIF formats are widely used for video conference


Y: 176 x 144

U: 88 x 72

V: 88 x 72

Sample Matlab code: `readyuv('foreman.qcif',176, 144, 1, 1);`

Color Space: YUV

YUV codes a luminance signal Y ' - brightness

Chrominance refers to the difference between a color and a reference white at the same luminance. → use color differences *U, V*:

$$U = B' - Y', \qquad V = R' - Y' \tag{4.27}$$

$$\begin{bmatrix} Y' \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.299 & -0.587 & 0.886 \\ 0.701 & -0.587 & -0.114 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} \tag{4.28}$$

For a gray clolor, $R' = G' = B'$, the luminance Y' equals to that gray, since 0.299+0.587+0.114 = 1.0. And for a gray ("black and white") image, the chrominance (U, V) is zero.

*Ref to **4.1.7** and **4.1.8***

(d) In the actual implementation *U* and *V* are rescaled to have a more convenient maximum and minimum.

(e) For dealing with composite video, it turns out to be convenient to contain the composite signal magnitude $Y' \pm \sqrt{U^2 + V^2}$ within the range –1/3 to +4/3. So *U* and *V* are rescaled:

$$U = 0.492111\ (B' - Y')$$

$$V = 0.877283\ (R' - Y') \qquad (4.29)$$

The chrominance signal = the composite signal *C*:

$$C = U \cdot \cos(\omega t) + V \cdot \sin(\omega t) \qquad (4.30)$$

(f) Zero is not the minimum value for *U*, *V* . *U* is approximately from blue (*U* > 0) to yellow (*U* < 0) in the RGB cube; V is approximately from red (*V* > 0) to cyan (*V* < 0).

(g) Fig. 4.18 shows the decomposition of a color image into its *Y*, *U*, *V* components. Since both *U* and *V* go negative, in fact the images displayed are shifted and rescaled.

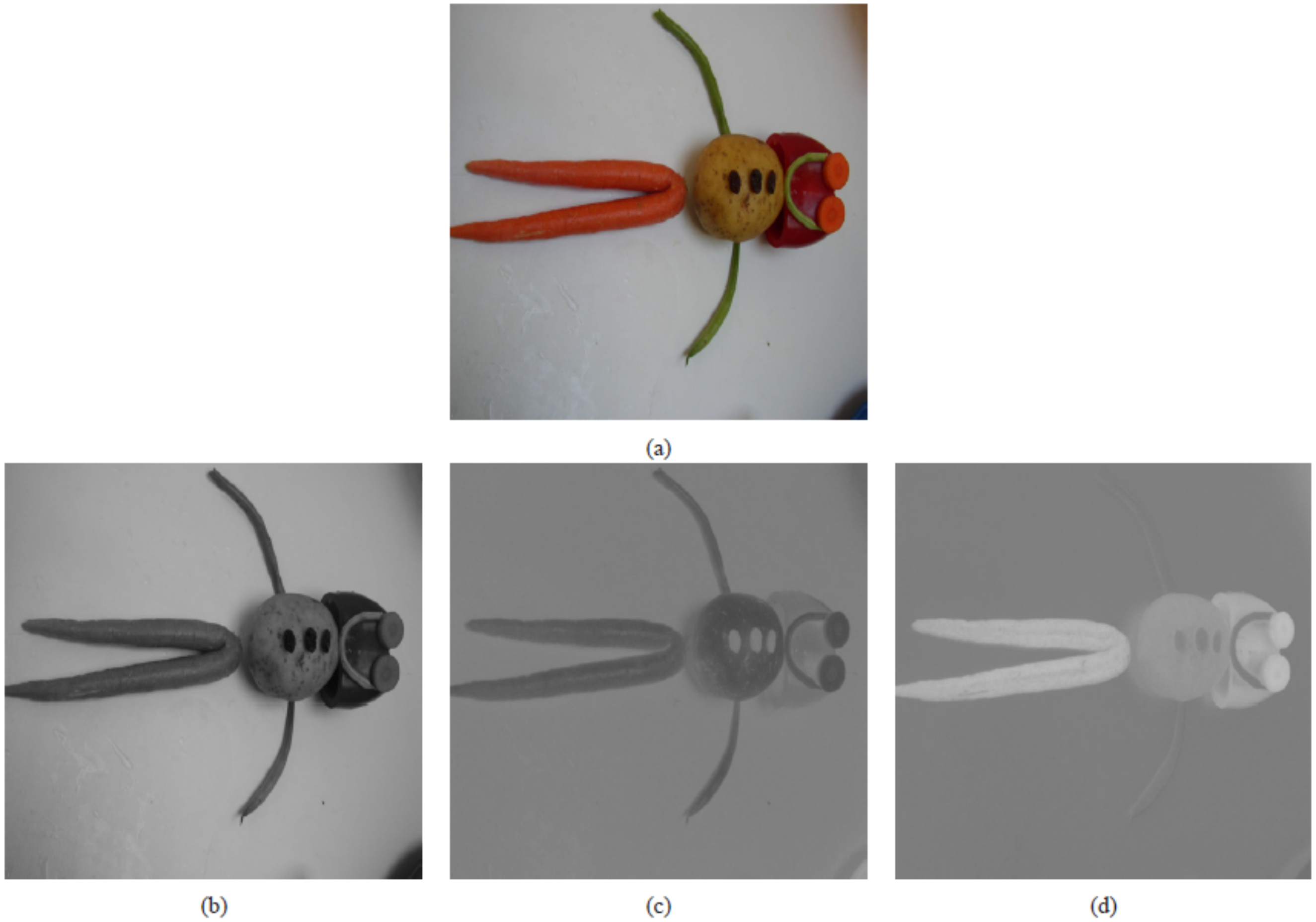


Fig. 4.18: Y 'UV decomposition of color image. Top image (a) is original color image; (b) is Y '; (c,d) are (U, V)

4.3.3 YIQ Color Model

- YIQ is used in NTSC color TV broadcasting. Again, gray pixels generate zero (I, Q) chrominance signal.

(a) *I* and *Q* are a rotated version of *U* and *V* .

(b) *Y '* in YIQ is the same as in YUV; *U* and *V* are rotated by 33°:

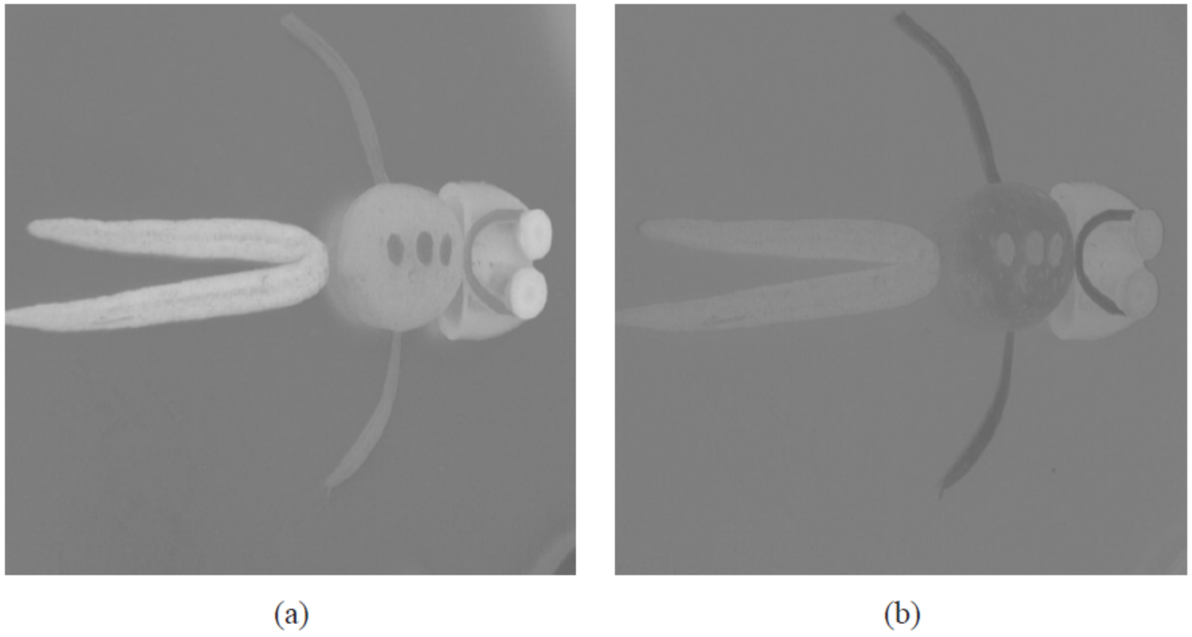
$$I = 0.492111(R' - Y') \cos 33° - 0.877283(B' - Y') \sin 33°$$

$$Q = 0.492111(R' - Y') \sin 33° + 0.877283(B' - Y') \cos 33° \tag{4.31}$$

(c) This leads to the following matrix transform:

$$\begin{bmatrix} Y' \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.595879 & -0.274133 & -0.321746 \\ 0.211205 & -0.523083 & 0.311878 \end{bmatrix} = \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} \tag{4.32}$$

(d) Fig. 4.19 shows the decomposition of the same color image as above, into YIQ components.



- **Fig.4.19**: (a) I and, (b) *Q* components of color image.

4.3.4 YCbCr Color Model

- The Rec. 601 standard for digital video uses another color space, YC_bC_r , often simply written YCbCr — closely related to the YUV transform.

  (a) YUV is changed by scaling such that C_b is U, but with a coefficient of 0.5 multiplying B'. In some software systems, C_b and C_r are also shifted such that values are between 0 and 1.

  (b) This makes the equations as follows:

$$C_b = ((B' - Y')/1.772) + 0.5 \qquad (4.33)$$

$$C_r = ((R' - Y')/1.402) + 0.5$$

  (c) Written out:

$$\begin{bmatrix} Y' \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.168736 & -0.331264 & 0.5 \\ 0.5 & -0.418688 & -0.081312 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} + \begin{bmatrix} 0 \\ 0.5 \\ 0.5 \end{bmatrix} \qquad (4.34)$$

d) In practice, however, Recommendation 601 specifies 8-bit coding, with a maximum *Y*′ value of only 219, and a minimum of +16. Cb and Cr have a range of ±112 and offset of +128. If *R*′, *G*′, *B*′ are floats in [0.. + 1], then we obtain *Y* ′, Cb, Cr in [0..255] via the transform:

$$\begin{bmatrix} Y' \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -37.797 & -74.203 & 112 \\ 112 & -93.786 & -18.214 \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \qquad (4.35)$$

e) The YCbCr transform is used in JPEG image compression and MPEG video compression.

Outline

- Color Science
- Color space: RGB
- Color Space: YUV and more
- Graphics/Image Data Types
  - Popular File Formats

Graphics/Image Data Types

- The number of file formats used in multimedia continues to proliferate.

Table 3.1: Adobe Premiere file formats.

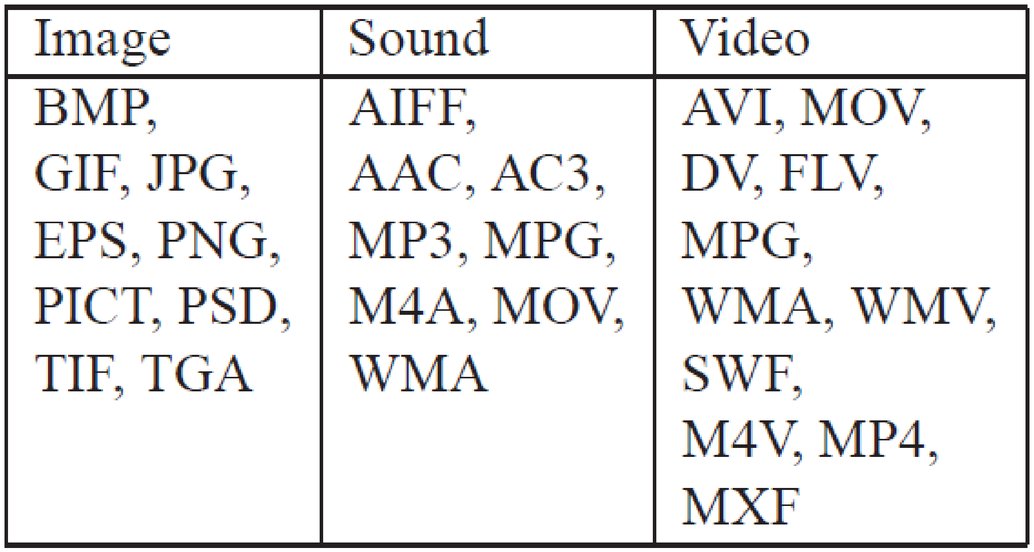


Image	Sound	Video
BMP, GIF, JPG, EPS, PNG, PICT, PSD, TIF, TGA	AIFF, AAC, AC3, MP3, MPG, M4A, MOV, WMA	AVI, MOV, DV, FLV, MPG, WMA, WMV, SWF, M4V, MP4, MXF

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. 1 shows a 1-bit monochrome image (called "Lena" by multimedia scientists — this is a standard image used to illustrate many algorithms).

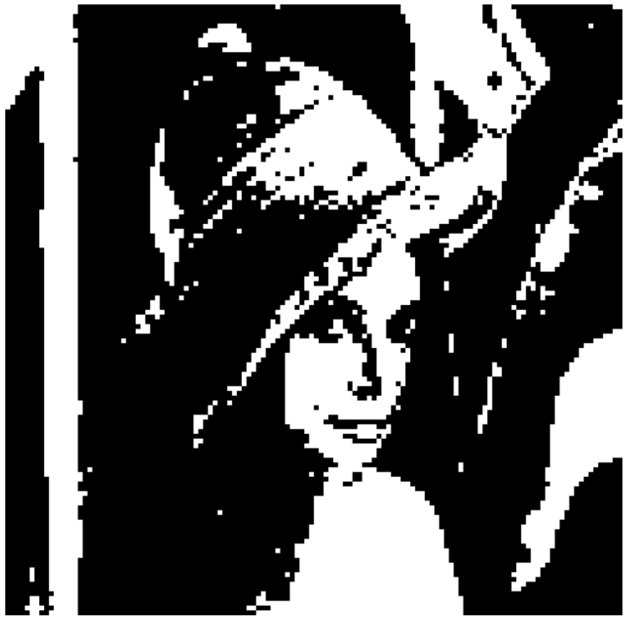


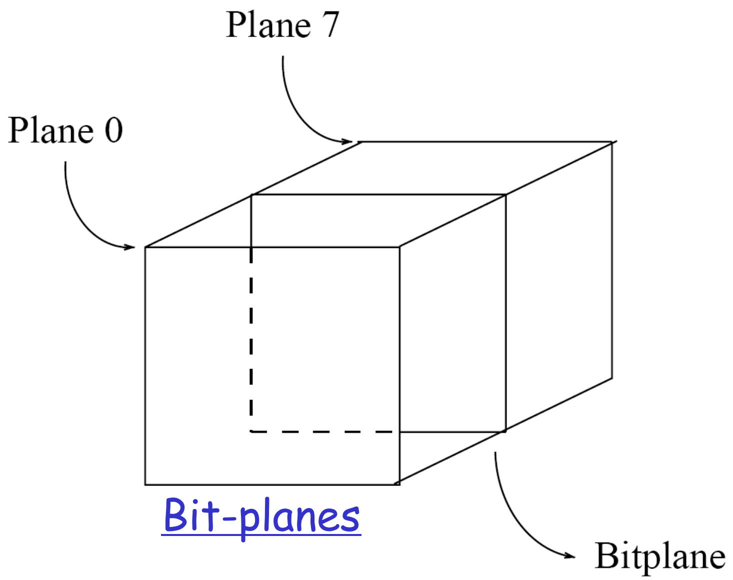
Fig. 1: Monochrome 1-bit Lena image.

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 x 1,200, whereas lower resolution might be 640 x 480.

8-bit Gray-level Images cont'd

- **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.
- 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.

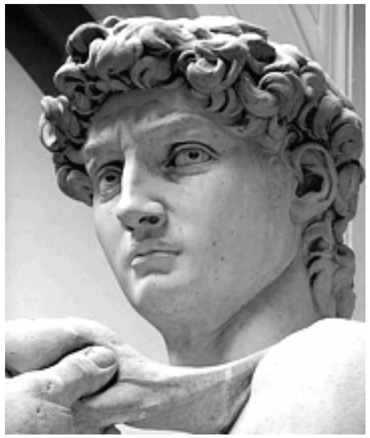
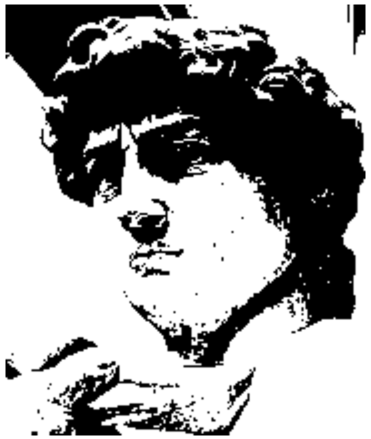


Bit-planes

Grayscale Image

- Each pixel -> a byte (a value between 0 to 255)
  - 640x480 grayscale image requires 300 kB of storage (640 *x* 480 = 307, 200).
- Print a grayscale image on Black/White newspaper (or laser printer) ?

  A note: A black/white newspaper displays black (one dot) or white (no dot) only, so does a black/white laser printer !

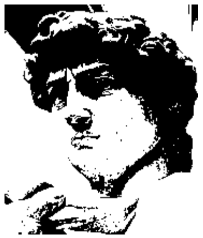



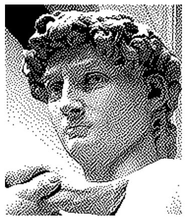
Grayscale Image

- Each pixel -> a byte (a value between 0 to 255)
  - 640x480 grayscale image requires 300 kB of storage (640 x 480 = 307, 200).
- Print a grayscale image on Black/White newspaper (or laser printer) ?
  - **Dithering** is used, which trades intensity resolution for spatial resolution to provide ability to print multi-level images on 2-level (1-bit) printers.

A note: A black/white newspaper displays black (one dot) or white (no dot) only, so does a black/white laser printer !

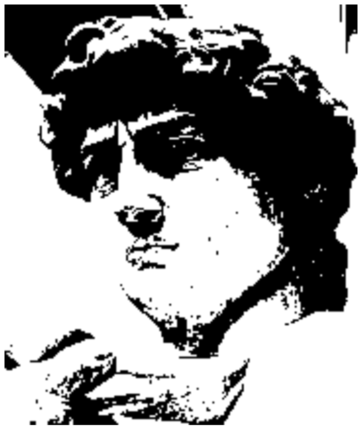
Grayscale Image Printing - Dithering

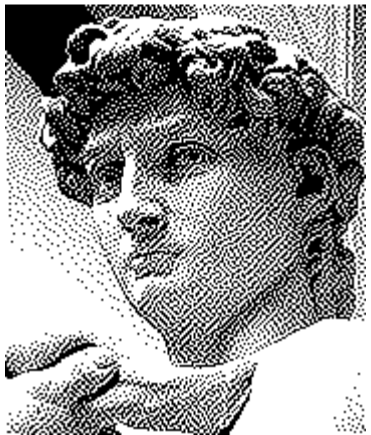




Bi-level only

With dithering





Dithering

- **Rationale:** calculate square 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.

- **Strategy:** Replace a pixel value by a larger pattern, say 2x 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**

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

Dithering cont'd

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.

Ordered Dithering

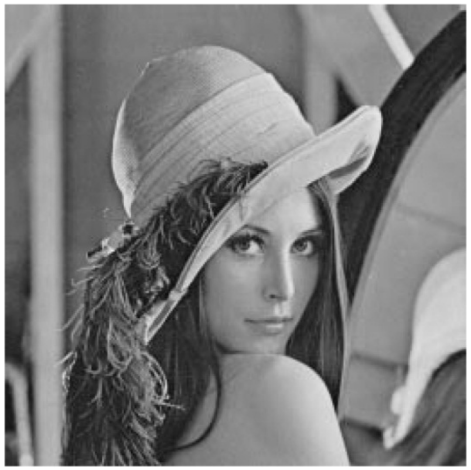
- A clever trick can get around this problem. Suppose we wish to use a larger, 4 x 4 dither matrix, such as

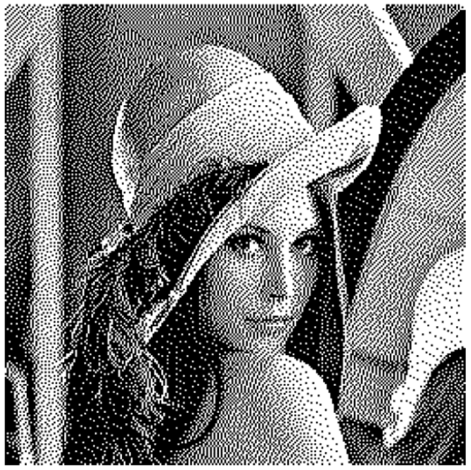
$$\begin{pmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{pmatrix}$$

- 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. 4 (a) shows a grayscale image of “Lena”. The ordered-dither version is shown as Fig. 4 (b), with a detail of Lena's right eye in Fig. 4 (c).

Dithering cont'd


(a)


(b)

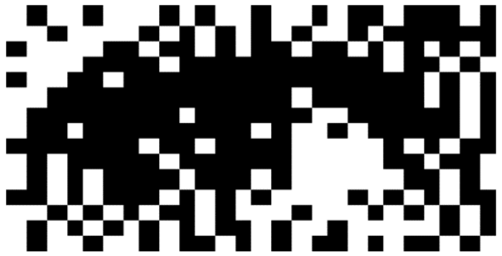

(c)

Fig. 4: Dithering of grayscale images.

(a): 8-bit grey image "lenagray.bmp". (b): Ordered dithered version of the image. (c): Detail of dithered version.

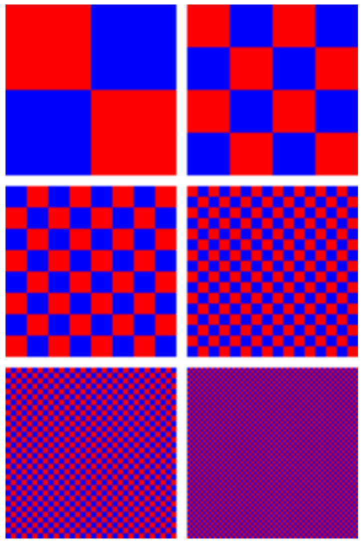
Dithering cont'd

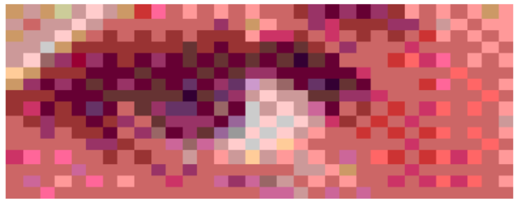
- Algorithm for ordered dither, with $n \times n$ dither matrix, is as follows:

```
BEGIN
    for $x = 0$ to $x_{max}$                  // columns
        for $y = 0$ to $y_{max}$              // rows
            $i = x \bmod n$
            $j = y \bmod n$
            // $I(x, y)$ is the input, $O(x, y)$ is the output,
            //$D$ is the dither matrix.
            if $I(x, y) > D(i, j)$
                $O(x, y) = 1;$
            else
                $O(x, y) = 0;$
END
```

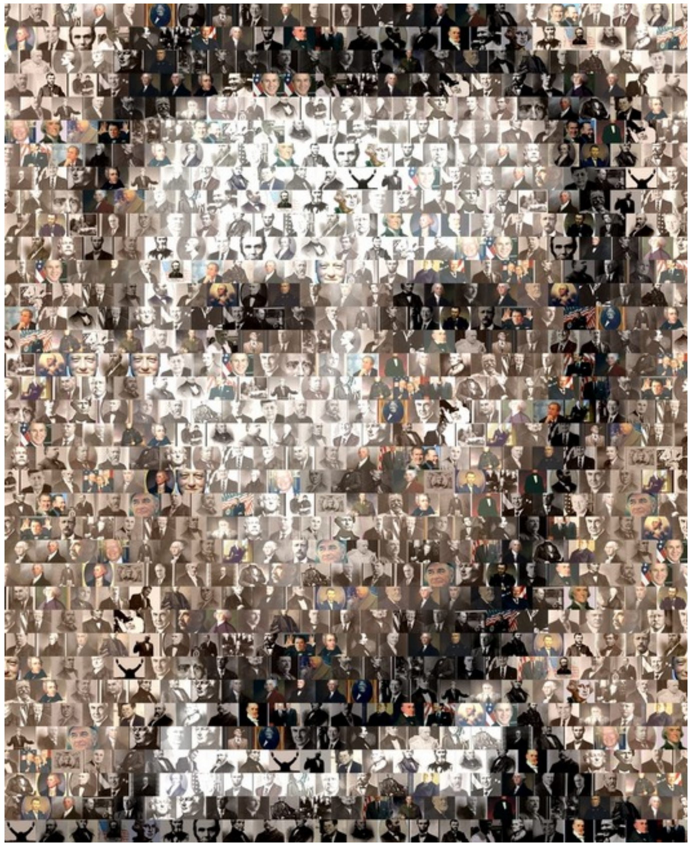
Colored Dithering

- Apply dithering to each color
- http://en.wikipedia.org/wiki/Dither





Colored Dithering



- pixisnap photos http://www.pixisnap.com/

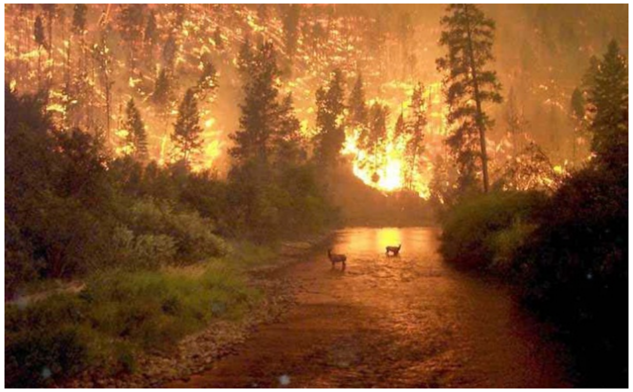
24-bit Color Images

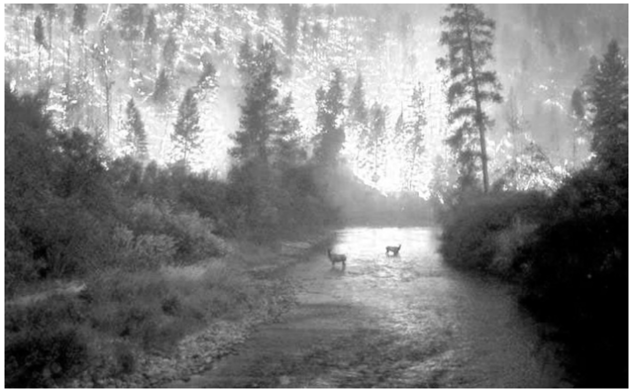
- In a color 24-bit image, each pixel is represented by three bytes, usually representing RGB.
  - This format supports 256x 256x 256 possible combined colors, or a total of 16,777,216 possible colors.
  - However such flexibility does result in a storage penalty: A 640x480 24-bit color image would require 921.6 kB of storage without any compression.

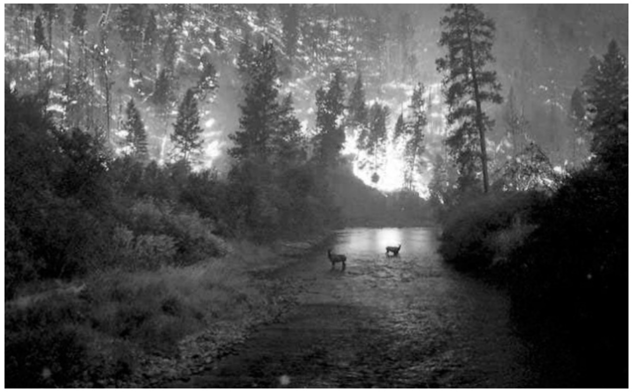
- **An important point**: many 24-bit color images are actually stored as 32-bit images, with the extra byte of data for each pixel used to store an *alpha* value representing special effect information (e.g., transparency).

- Fig. 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.

24-bit Color Images


(a)


(b)


(c)

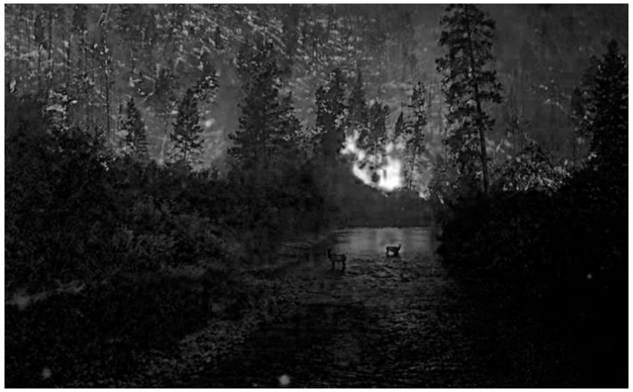

(d)

Fig. 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

Beyond 24-bit Color 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)
  - 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 cancer.
  - in satellite imaging, use high-Depth 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)

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. Why ?

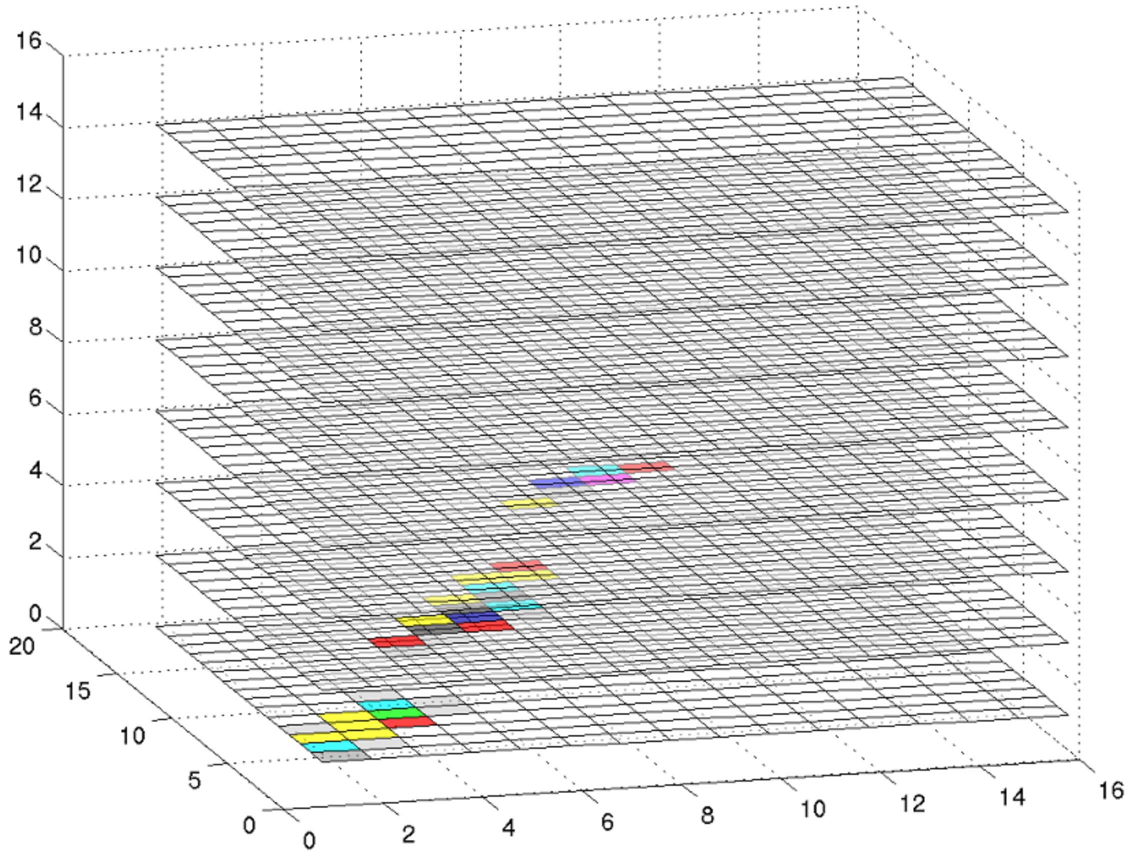


Fig. 6: 3-dimensional histogram of RGB colors in “forestfire.bmp”.

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 **(color index) 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.

8-bit Color Images cont'd

Example of 8-bit color image.

- Note the great savings in space for 8-bit images, over 24-bit ones: a 640x480 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).

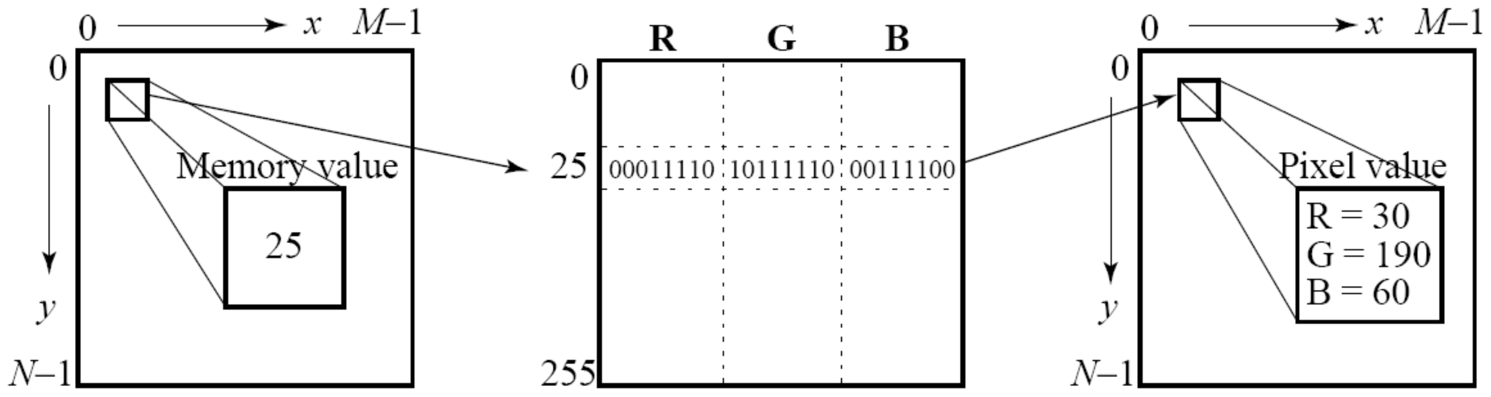
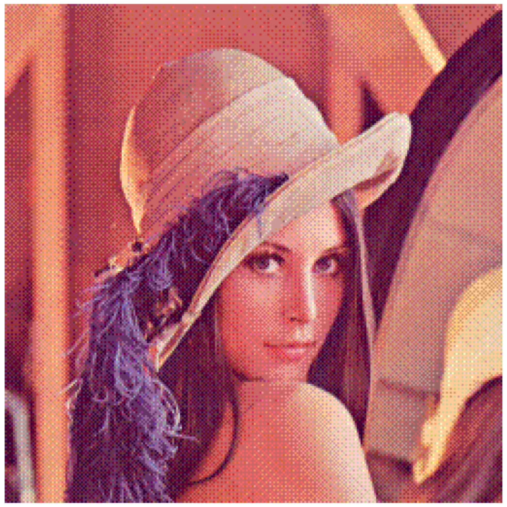


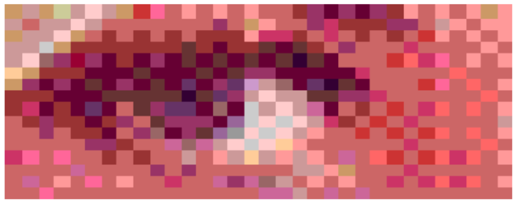
Fig. 8: Color LUT for 8-bit color images.

Color Look-up Tables (LUTs) cont'd

- (a): A 24-bit color image of "Lena"
- (b): The same image reduced to only 5 bits via dithering
- (c): A detail of the left eye

(a)


(b)


(c)

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.

Problem

- 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 be quite annoying perceptually.
- Imaging a smoothly changing color suddenly divided into two range
- E.g.
  - Use 3 bits for R, then the original value range from 0~255 with interval 1
  - Now become 0~255 with interval 32(256/2^3)
  - So two color RGB(31,128,128), RGB(32,128,128) would become RGB(0,128,128), RGB(32,128,128)

- **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.

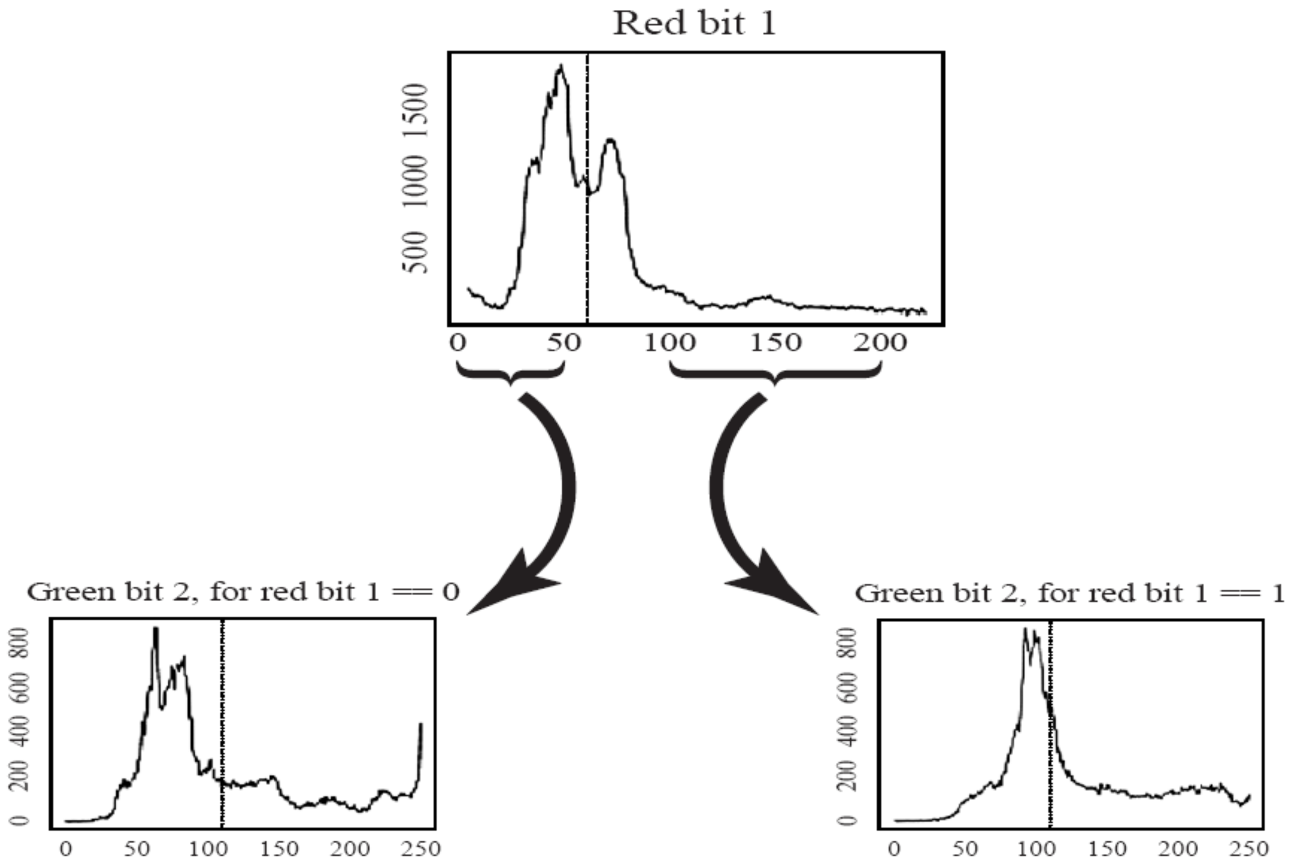
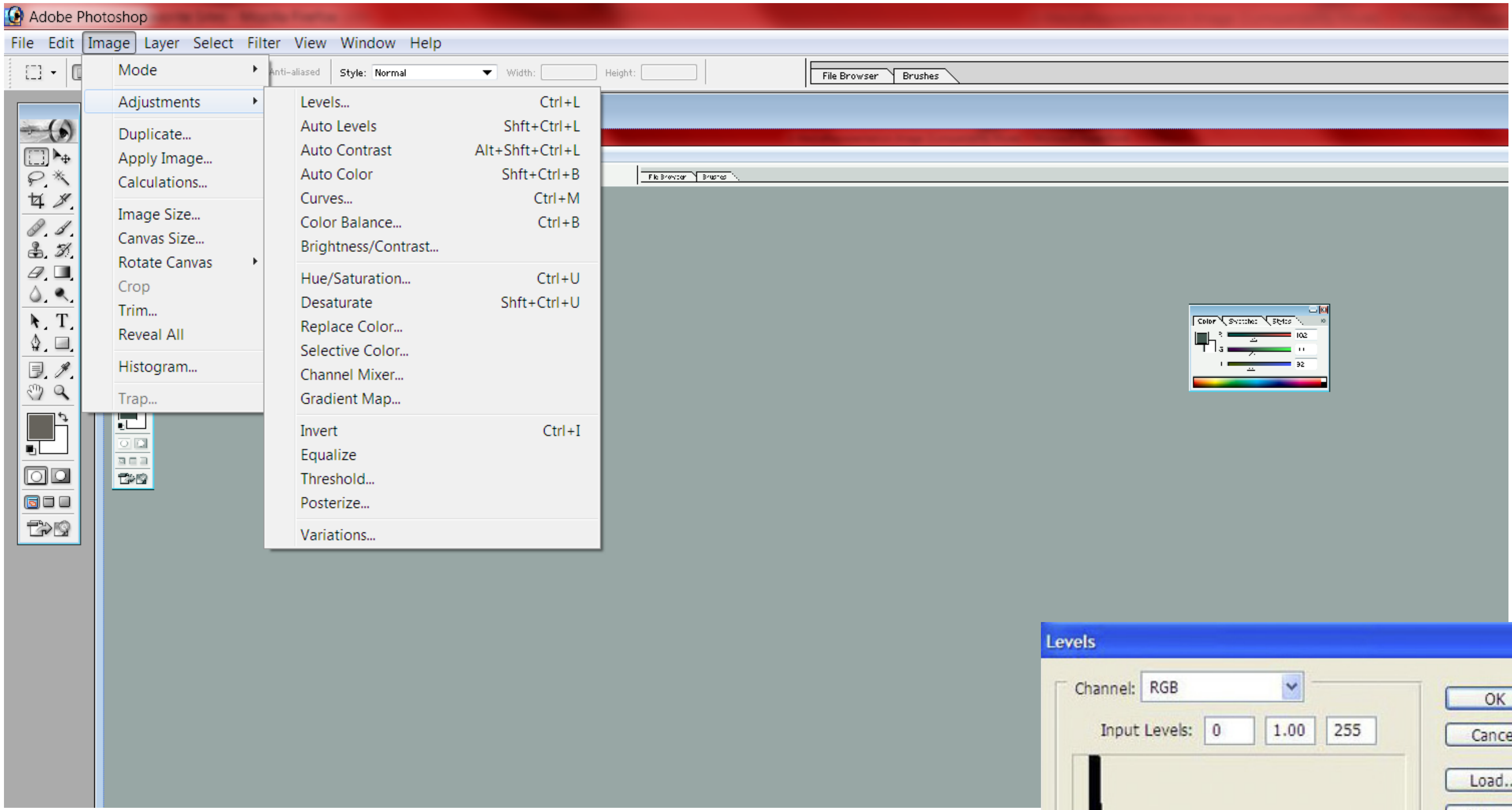
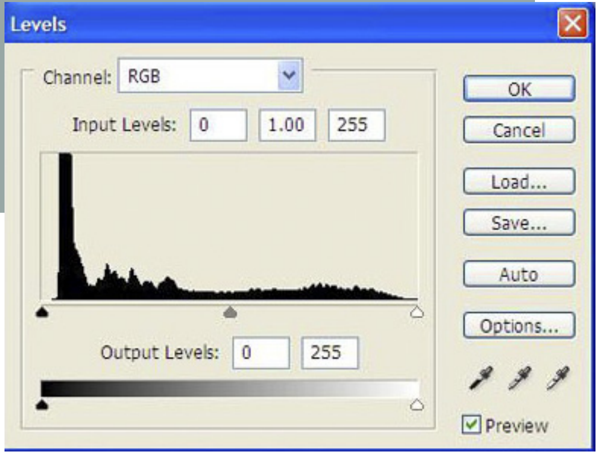


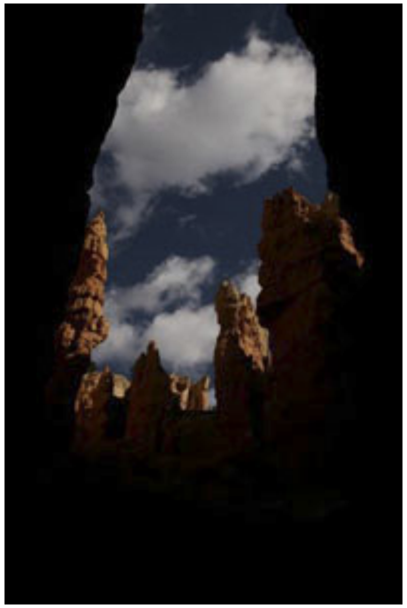
Fig. 3.11: Histogram of R bytes for the 24-bit color image “forestfire.bmp” results in a “0” bit or “1” bit label for every pixel. For the second bit of the color table index being built, we take R values less than the R median and label just those pixels as “0” or “1” according as their G value is less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

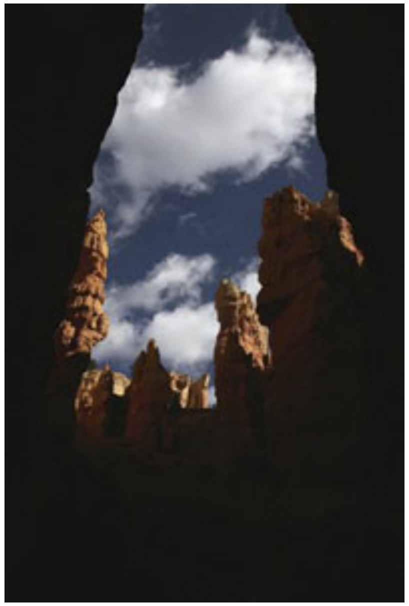
More about Histogram (and the power of digitization)

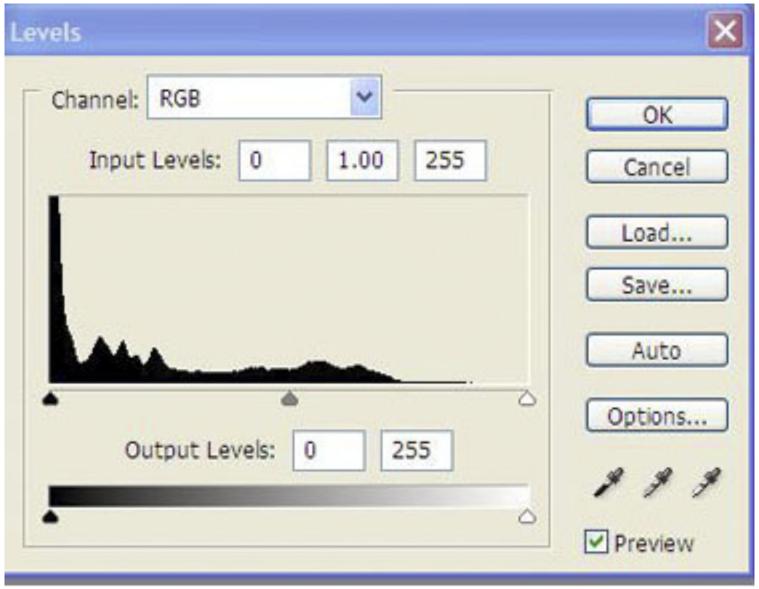


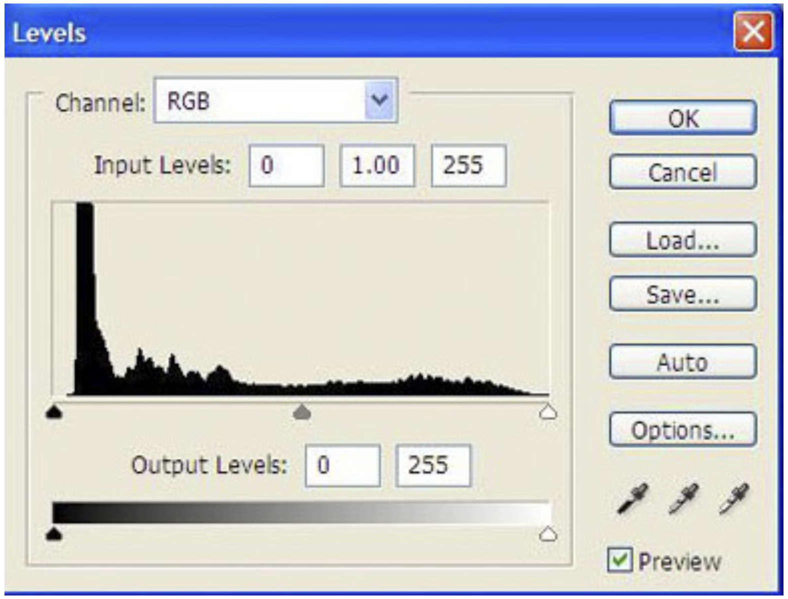


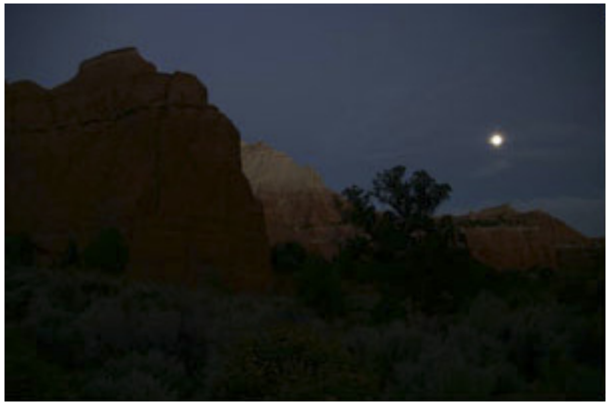
More about Histogram (and the power of digitization)

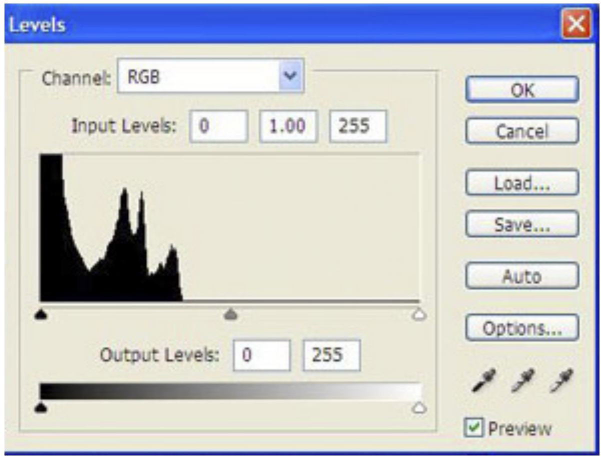






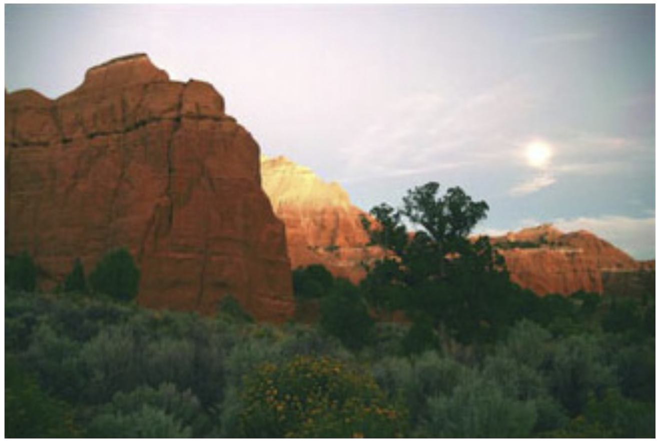


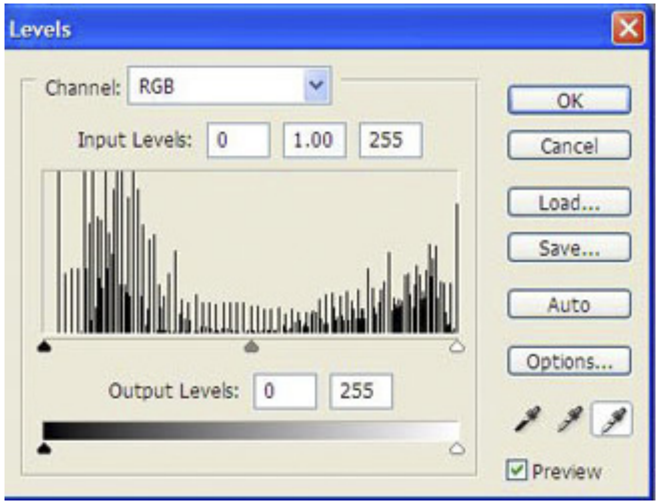




Input Levels: 0 1.00 255

Output Levels: 0 255





Input Levels: 0 1.00 255

Output Levels: 0 255

Does Photoshop Auto Levels Reveal Mars True Colors?

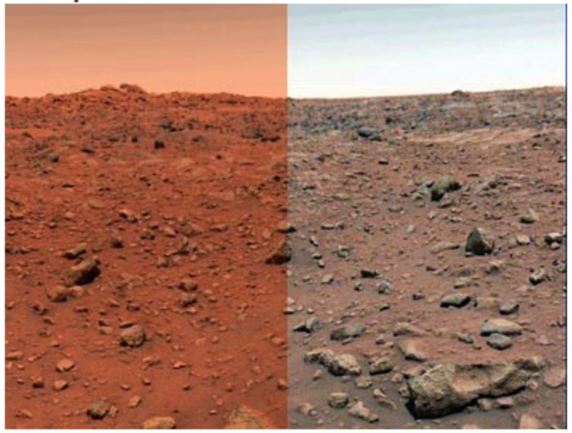
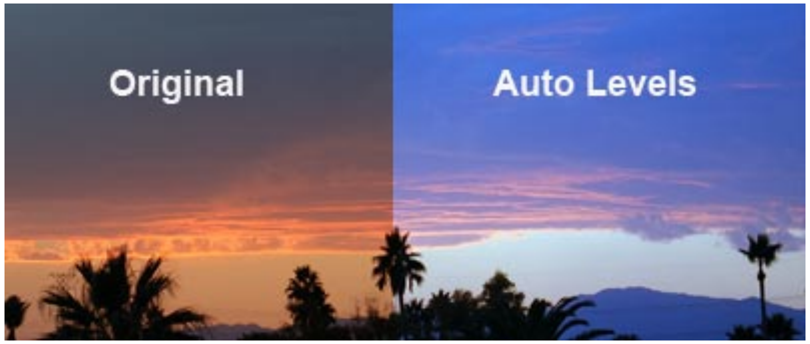
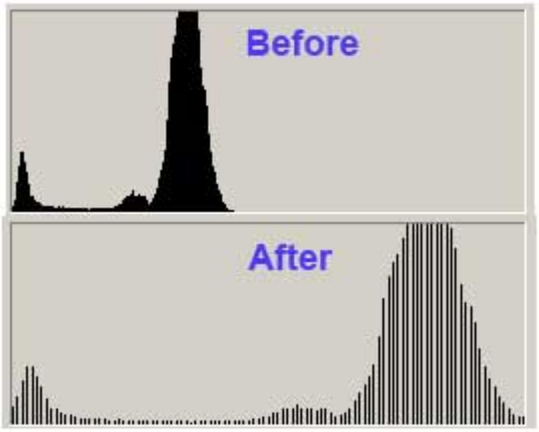


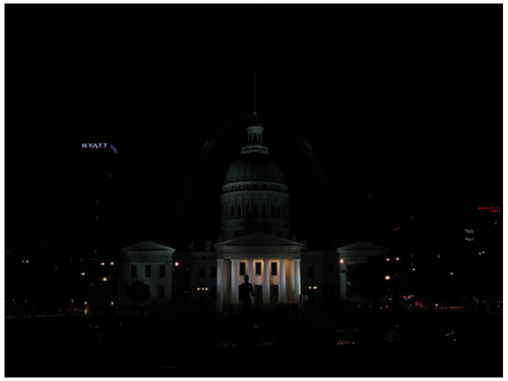
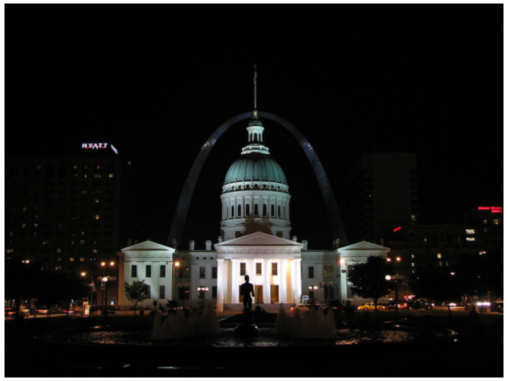
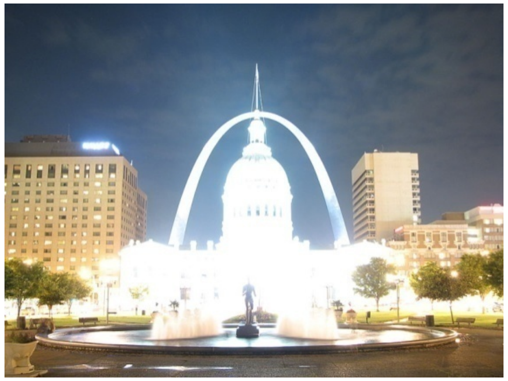
Figure 1. Viking 1 Lander color view of the martian surface.
Left: Mars as NASA shows it.
Right: Mars as Hoagland and company believe it to be using Photoshop Auto Levels.

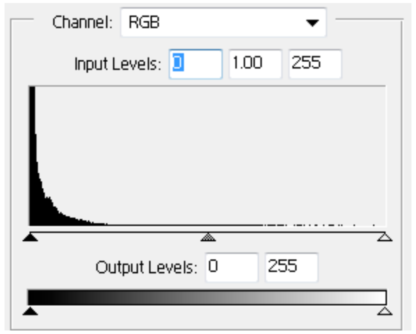


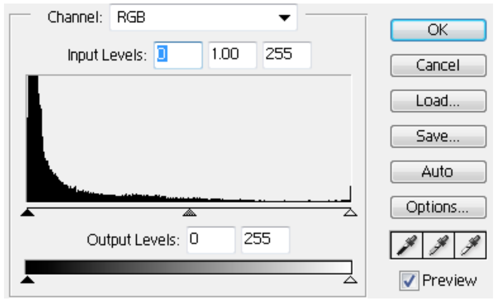


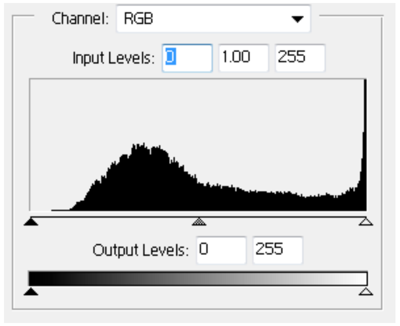
High Dynamic Range (HDR) Image

- http://en.wikipedia.org/wiki/High_dynamic_range_imaging

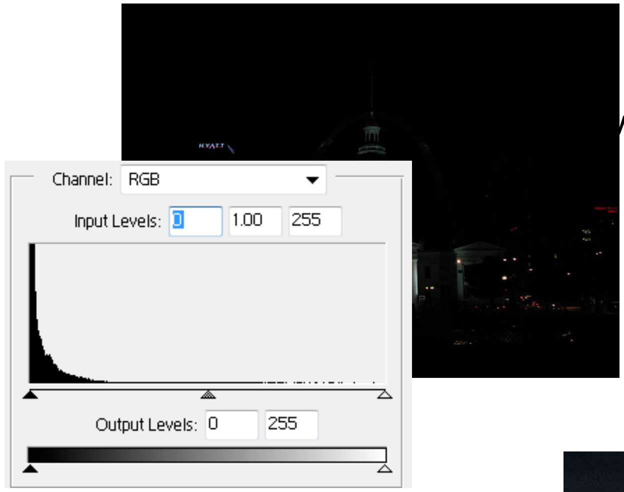


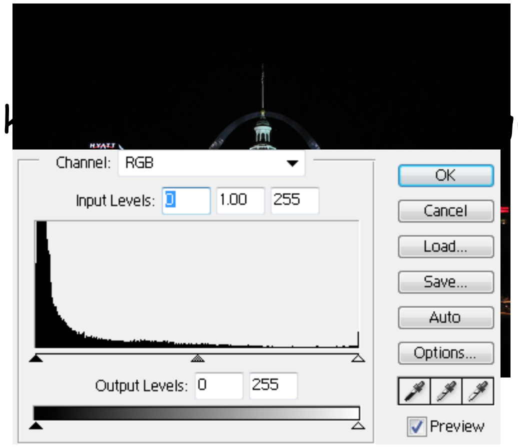


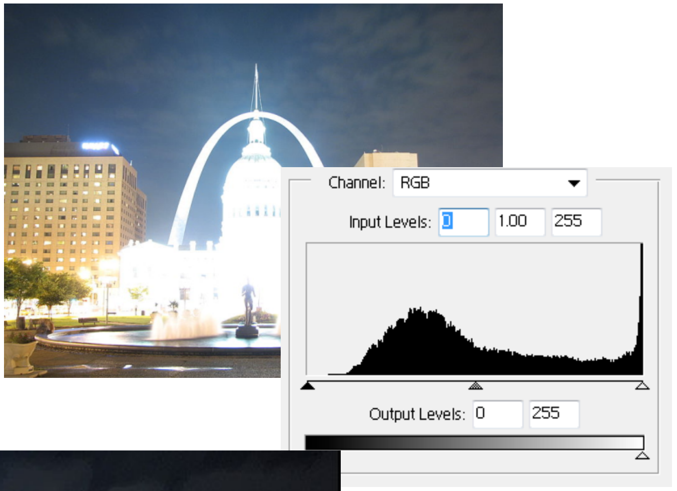


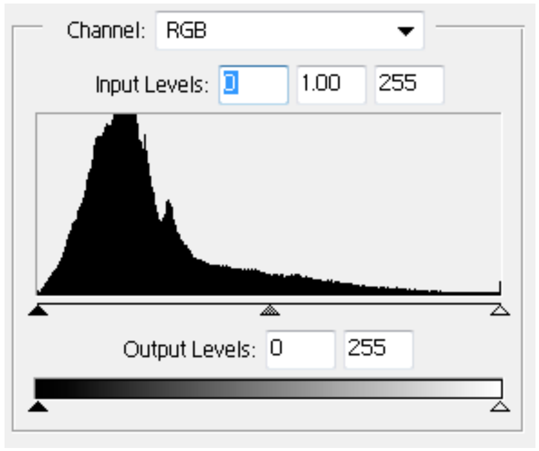
High Dynamic Range (HDR) Image

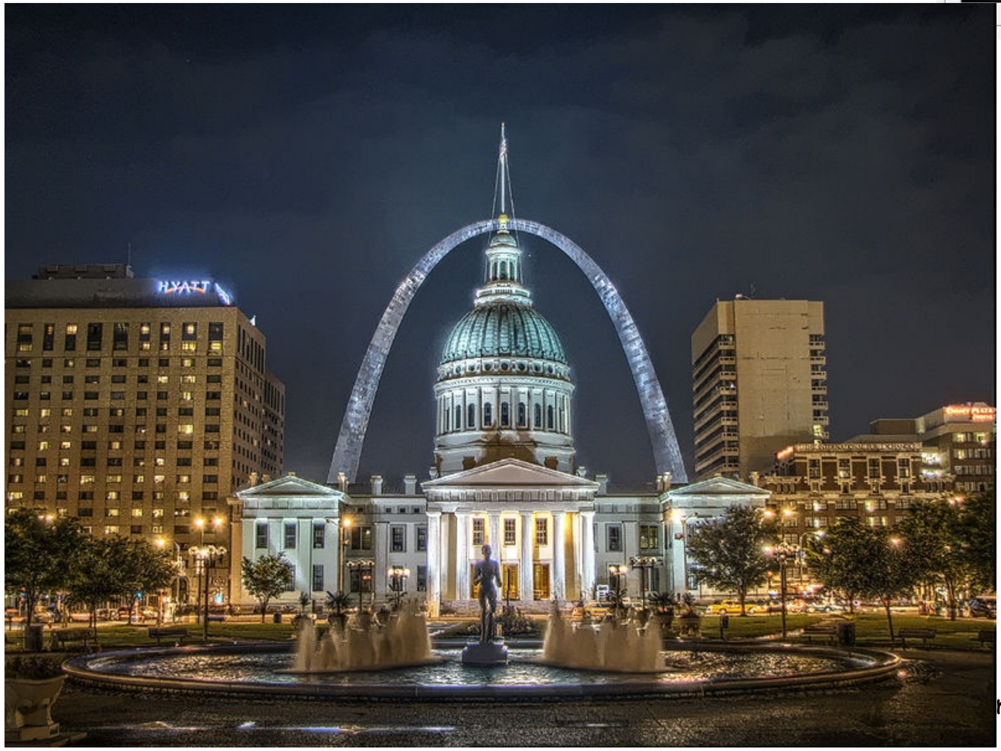
wik ing


Channel: RGB
Input Levels: 0 1.00 255
Output Levels: 0 255

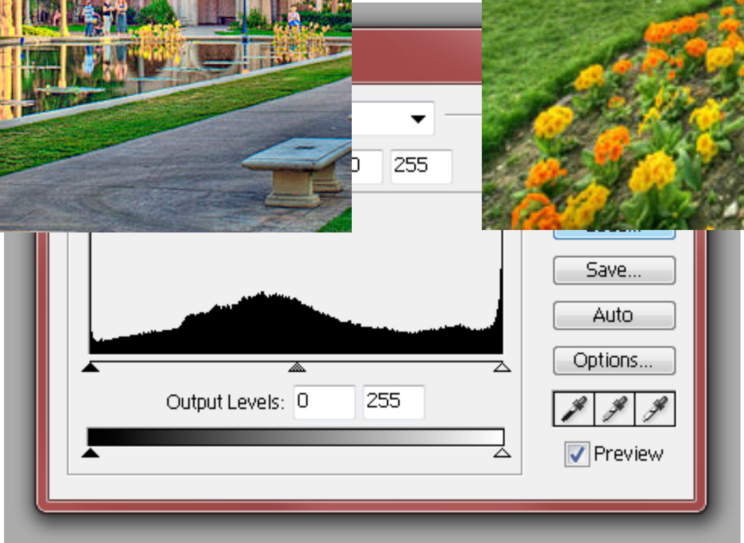

Channel: RGB
Input Levels: 0 1.00 255
Output Levels: 0 255


Channel: RGB
Input Levels: 0 1.00 255
Output Levels: 0 255


Channel: RGB
Input Levels: 0 1.00 255
Output Levels: 0 255



High Dynamic Range (HDR) Image



0 255

Output Levels: 0 255

Outline

- Color space: RGB
- Color Space: YUV and more
- Graphics/Image Data Types
  - Popular File Formats

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.

GIF

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

- General file format of a GIF87 standard file

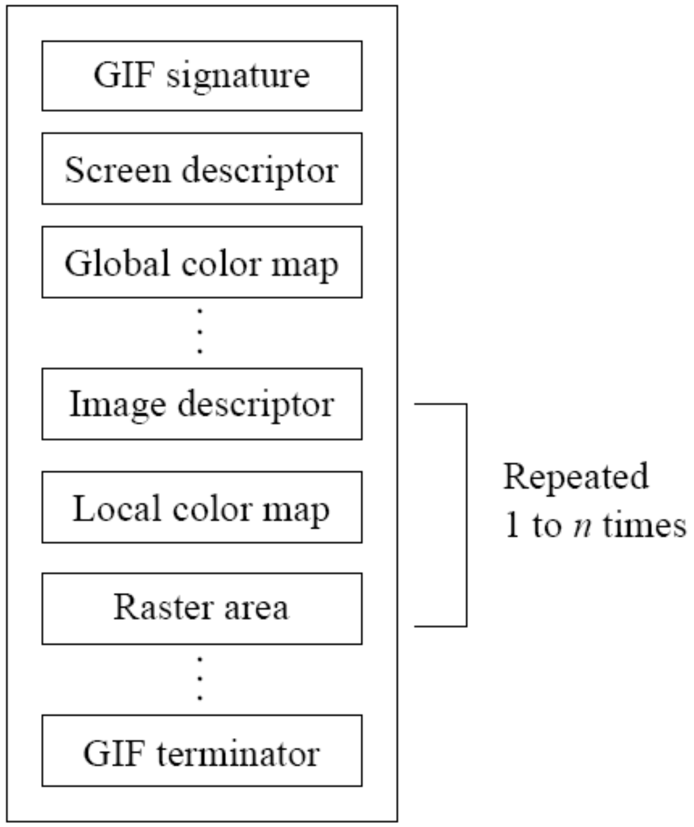


Fig. 12: GIF file format.

- **Screen Descriptor** comprises a set of attributes that belong to every image in the file.

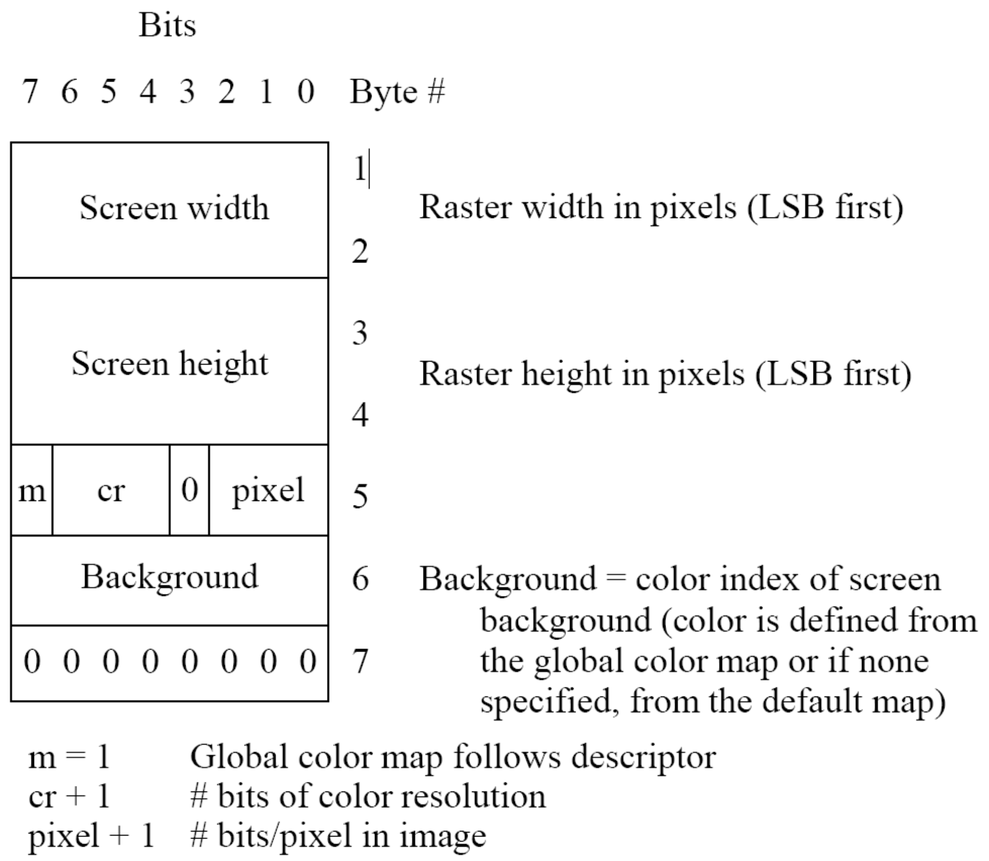

Bits
7 6 5 4 3 2 1 0
Byte #
Screen width
1
2
Raster width in pixels (LSB first)
Screen height
3
4
Raster height in pixels (LSB first)
m
cr
0
pixel
5
Background
6
Background = color index of screen background (color is defined from the global color map or if none specified, from the default map)
0 0 0 0 0 0 0 0
7
m = 1 Global color map follows descriptor
cr + 1 # bits of color resolution
pixel + 1 # bits/pixel in image

Fig.13: GIF screen descriptor in GIF87

- **Color Map** is set up in a very simple fashion as in Fig. 14. However, the actual length of the table equals $2^{(pixel+1)}$ as given in the Screen Descriptor.

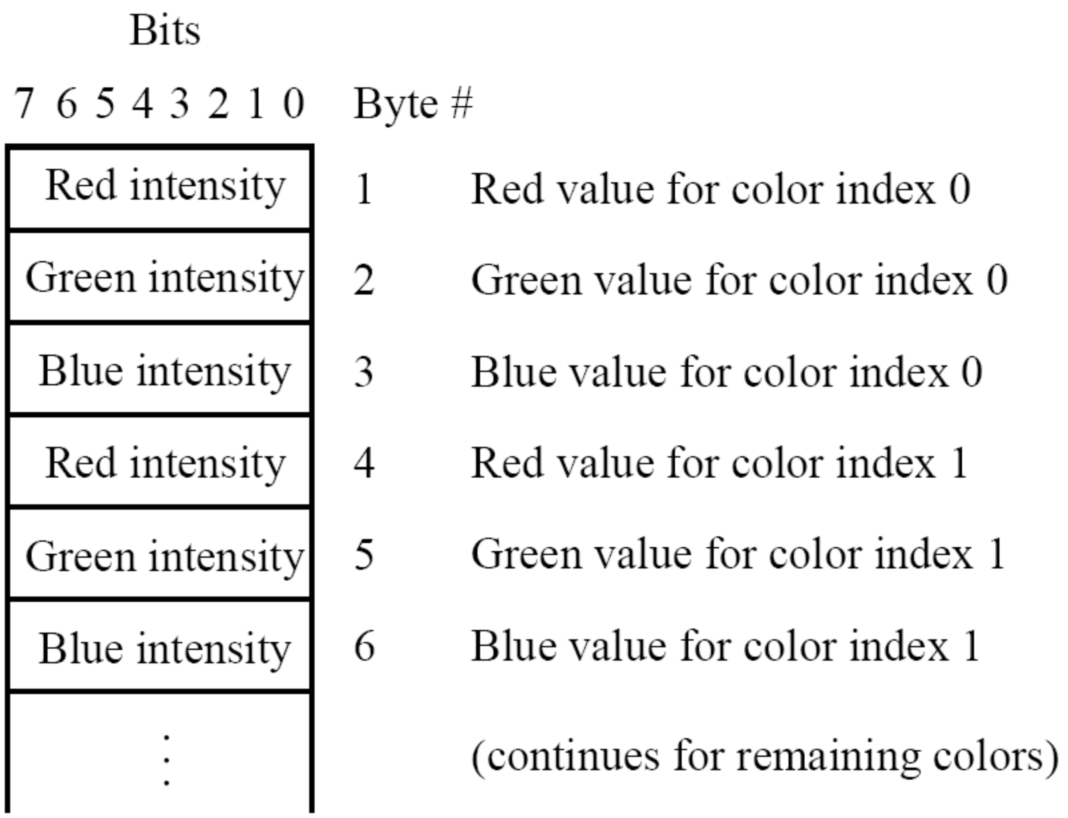
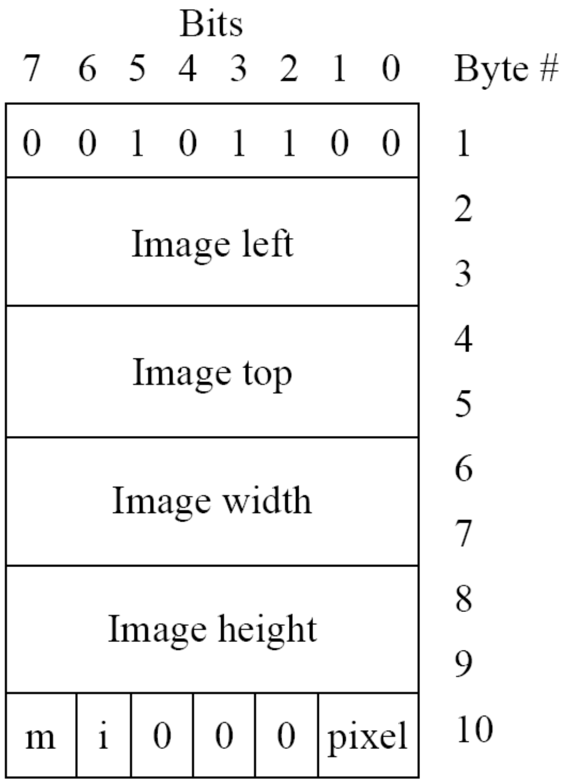

Bits
7 6 5 4 3 2 1 0
Byte #
Red intensity
1
Red value for color index 0
Green intensity
2
Green value for color index 0
Blue intensity
3
Blue value for color index 0
Red intensity
4
Red value for color index 1
Green intensity
5
Green value for color index 1
Blue intensity
6
Blue value for color index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

Fig. 14: GIF color map in GIF87

- Each image in the file has its own **Image Descriptor**



Byte #	
1	Image separator character (comma)
2, 3	Start of image in pixels from the left side of the screen (LSB first)
4, 5	Start of image in pixels from the top of the screen (LSB first)
6, 7	Width of the image in pixels (LSB first)
8, 9	Height of the image in pixels (LSB first)

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. 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

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. 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
  ```

JPEG

- **JPEG**: The most important current standard for image compression.
  -- Joint Photographic Experts Group
- 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. 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.
- More on later …

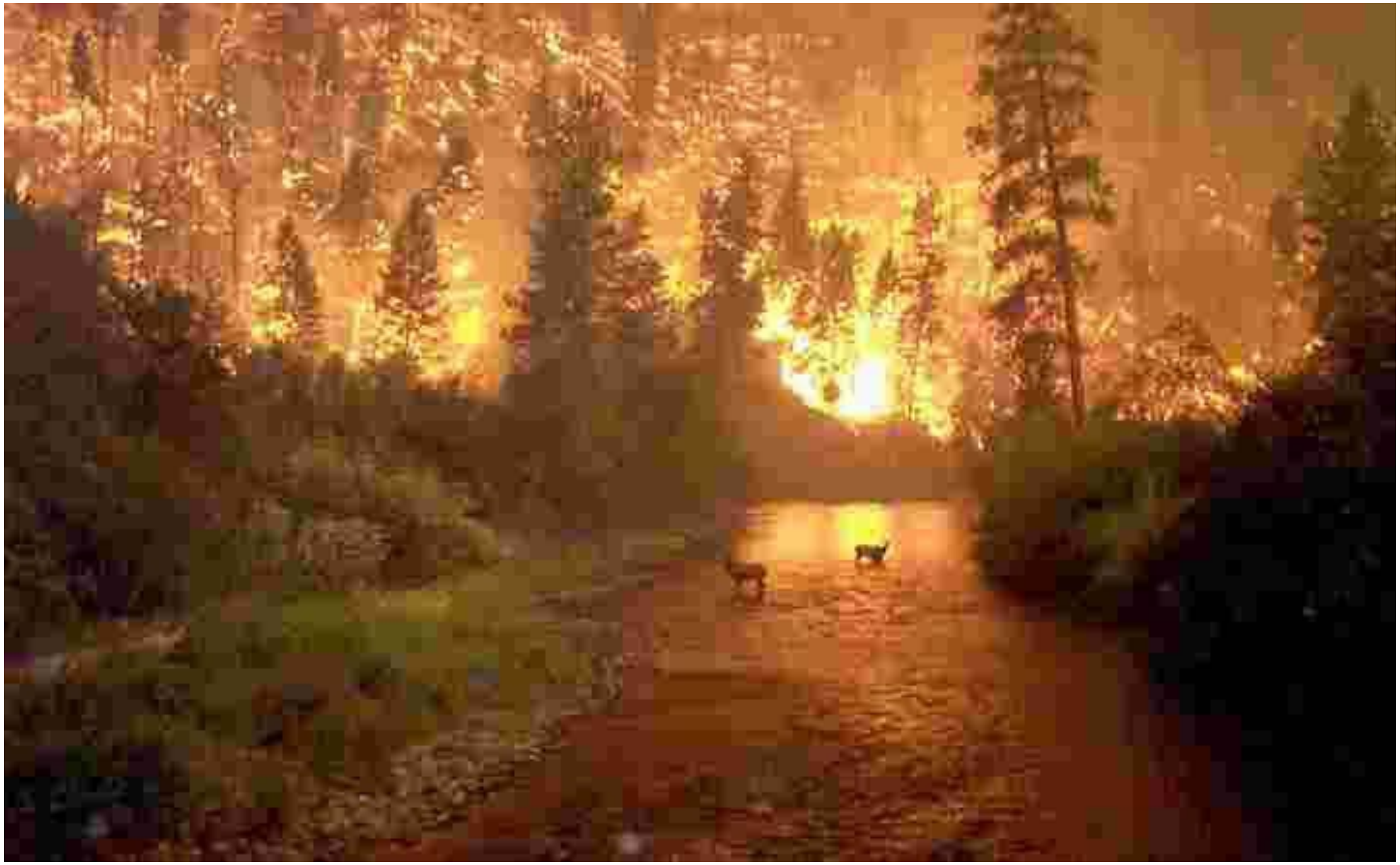


Fig. 17: JPEG image with low quality specified by user.

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 8×8 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 new 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).

Graphics Animation Files

- A few dominant formats aimed at storing graphics animations (i.e., series of drawings or graphic illustrations) as opposed to video (i.e., series of images).
- **Difference**: animations are considerably less demanding of resources than video files.
- 1. FLC is an animation or moving picture file format; it was originally created by Animation Pro. Another format, FLI, is similar to FLC.
- 2. GL produces somewhat better quality moving pictures. GL animations can also usually handle larger file sizes.
- 3. Many older formats: such as DL or Amiga IFF files, Apple Quicktime files, as well as animated GIF89 files.

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. GL bit-mapped images can be included in output files.

  3. Encapsulated Postscript files 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 begun to supersede or at least parallel 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: BMP**
  - Major system standard graphics file format for Microsoft Windows, used in Microsoft Paint and other programs
  - Many sub-variants within the BMP standard

- **Microsoft Windows: WMF (Windows Metafile)**
  - Native vector file format for the Windows operating environment:
  - Features

    1. Consist of a collection of GDI (Graphics Device Interface) function calls, also native to the Windows environment.

       -- e.g., PlayMetaFile() function to render

    2. Ostensibly device-independent and size unlimited

Many more... (check "save as" in Photoshop)

Further Exploration

- Chapter 3
- Chapter 4.1.1-4.1.6 4.3