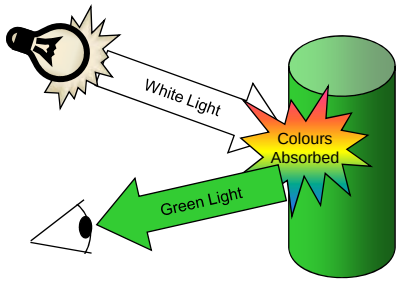
How Do We See Things?

- An object is seen/recognized from the visible light reflected from it
- Without light, we can not see
- If white light shines into a green object, most wavelengths are absorbed, except green light is reflected from the object
- Do not forget that white light can be decomposed into seven colors



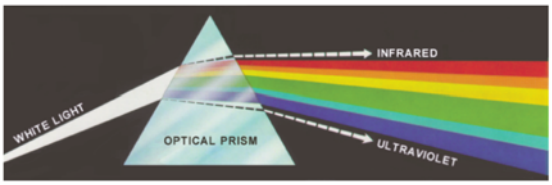


Image Acquisition

- Image acquisition is very similar to how we see things
  - ***illuminating*** the scene by an ***energy source***
  - ***recording*** the ***reflected*** or ***transmitted energy*** using sensors

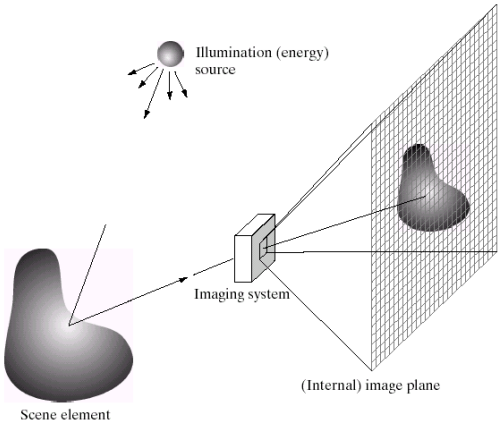
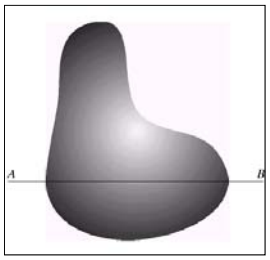
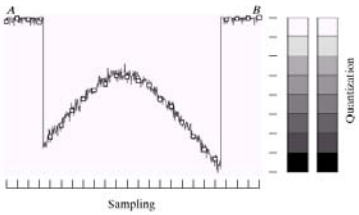


Image Acquisition

- A digital sensor can only return a ***discrete*** set of ***energy levels***
- ***Digitization (a.k.a. quantization)*** is the process of converting a continuous ***analogue*** signal into a ***digital representation*** of this signal
- At the end, you get some numbers, not colors





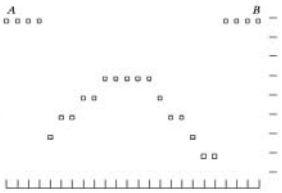
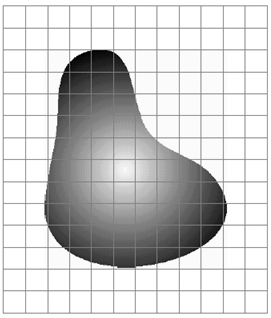
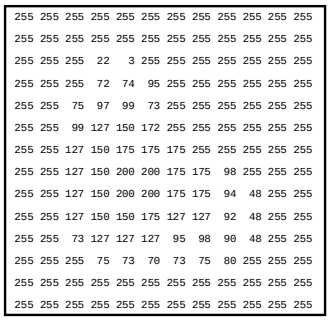


Image Acquisition

- Image viewers convert these numbers into color before displaying them
- A digital image is always an ***approximation*** of a real world scene





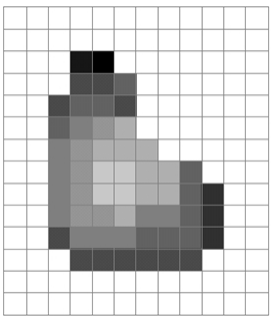


Image Acquisition

- Acquiring a color image is no more than using
  - Three filter that are sensitive to red, green, and blue
  - Three sensors to measure the intensity of the output of each filter
- The results are three ***monochrome*** images
  - The intensities of each image are proportional to the responses of one of the filters

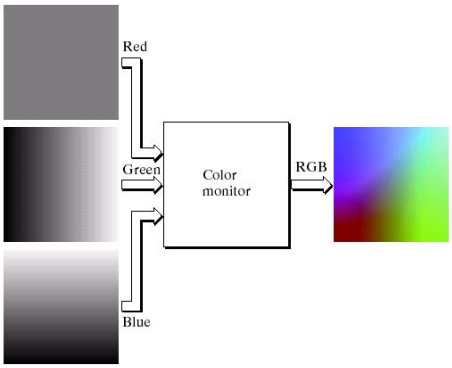
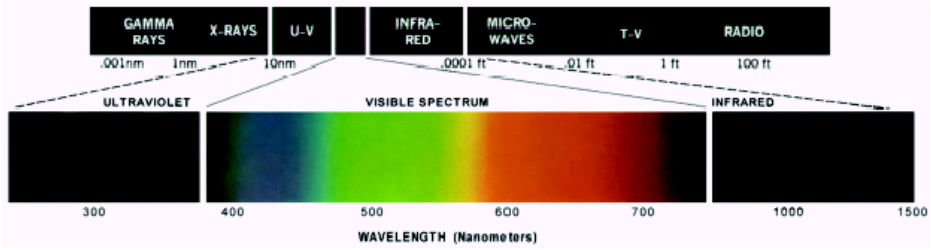


Image Acquisition

- In the case of humans, the energy source is the visible light, which is a kind of electromagnetic wave
- There are various electromagnetic waves, for example:



- ***Based on the wavelength of the energy source used, you get various image modalities***
- ***For each electromagnetic waves, we need a special sensor to record and quantize reflected energy (humans can not see any electromagnetic waves, other than the visible spectrum)***

Image Acquisition

- In a vacuum, electromagnetic waves travel with a speed equal to the speed of the light (300,000,000 meters/second)
- The relation between wavelength, frequency and speed is

  Speed = *Wavelength* × *Frequency*
- High frequencies mean
  - ☐ shorter wavelengths
  - ✓ more details (*a wave can not capture details smaller than its length*) -- your fingertip can not recognize details smaller than its size
  - × Less penetration

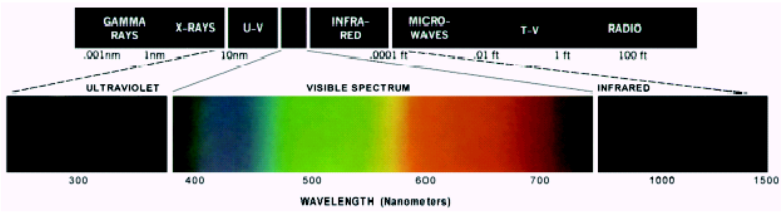
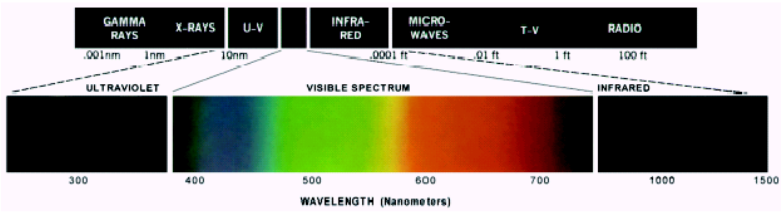
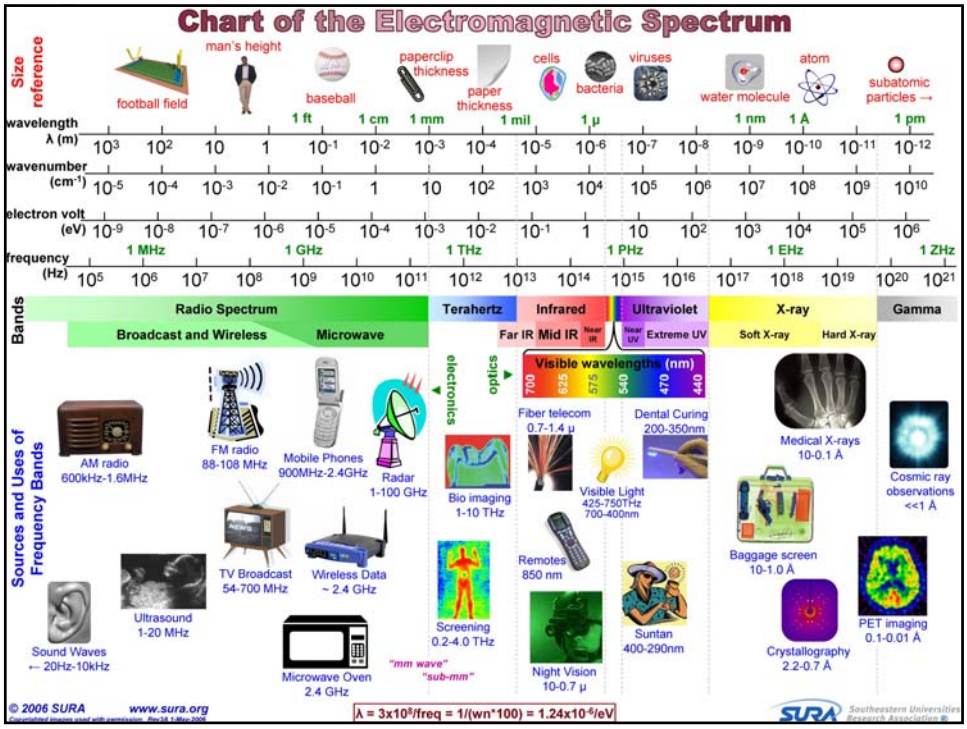


Image Acquisition

- Let us revisit the medical image modalities that we know
  - ☐ X-ray
  - ☐ Angiography
  - ☐ Video Fluoroscopy
  - ☐ Computed Axial Tomography (CAT), or simply Computed Tomography (CT)
  - ☐ Magnetic Resonance Imaging (MRI)
  - ☐ Ultrasound
  - ☐ Nuclear medicine
- *The right modality should be used to visualize what you want to see*





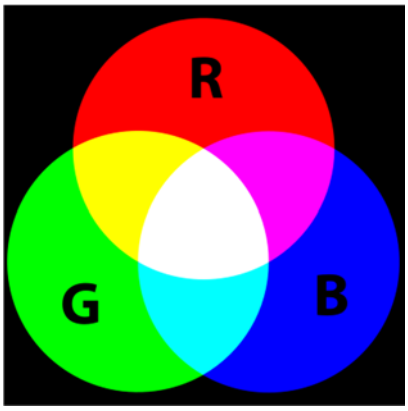
Color Models

- There are many ways to numerically represent colors
- A system for representing colors is called a *color model*
- Color models are usually designed to take advantage of a particular type of display devices
- Color models include
  - RGB color model
  - YC_bC_r color model
  - CMYK color model
  - HSI color model

RGB Color Model

- RGB is perhaps the most widely used color system in imaging today
- In RGB, colors are composed of three component values that represent the relative intensities of red, green, and blue
- RGB is an *additive* system in which varying amounts of the red, green and blue colors are added to black color to produce new color
- In RGB images, each pixel is represented as a color triplet, i.e., three numerical values in the form (red, green, blue)
- For 24-bit color,
  - (0, 0, 0) represents black
  - (255, 255, 255) represents white

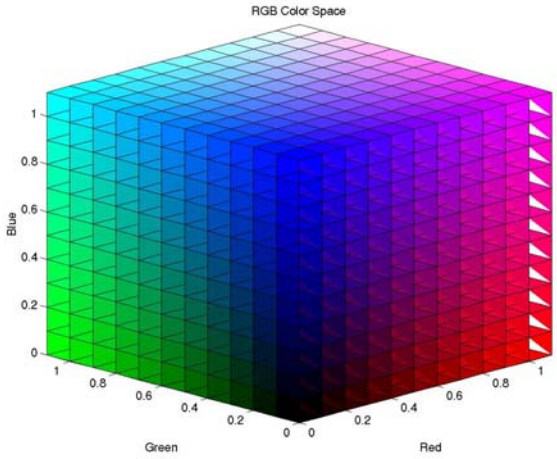
RGB Color Model



Additive color system

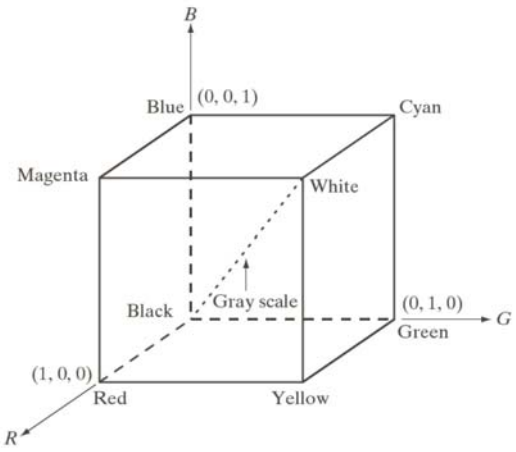
RGB Color Model

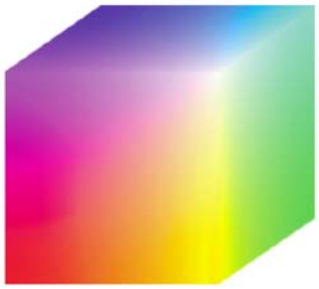
- How many different colors do we have here?



RGB Color Model

- If we divided each of the main three axes to 256 quantization levels, how many colors will we have in total?

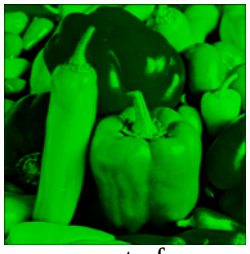




RGB Color Model

Peppers image

The R component of peppers image



The G component of peppers image

The B component of peppers image

RGB Color Model

peppers RGB image

peppers GRB image

peppers BRG image

peppers **R**BG image

peppers GBR image

peppers B**G**R image

YC_bC_r Color Model

- The YC_bC_r is another *additive* color system, which is similar to the one used in television set that allows color images to be compatible with black and white sets
- The Y component represents the intensity of the image; hence it is called the *luminance* component
- The C_b and C_r components specify the blueness and redness of the image, respectively; hence they are called *chrominance* components
- ***Q: Where is the green representation (green component) in the YC_bC_r color model?***

YC_bC_r Color Model

- To convert between RGB and YC_bC_r spaces, the following linear transformations may be used

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 0.2990 & 0.5870 & 0.1140 \\ -0.1687 & -0.3313 & 0.5000 \\ 0.5000 & -0.4187 & -0.0813 \end{bmatrix} \times \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1.40200 \\ 1 & -0.34414 & -0.71414 \\ 1 & 1.72200 & 0 \end{bmatrix} \times \begin{bmatrix} Y \\ C_b - 128 \\ C_r - 128 \end{bmatrix}$$

- *If $R \in \{0,1, ..,255\}$, $G \in \{0,1, ..,255\}$, and $B \in \{0,1, ..,255\}$, what is the range of Y, C_b, and C_r?*
- *What is the value of the R, G, and B, if Y = 0, C_b= 228, and C_r = 228?*
- *How come G is negative?* **F: Domain → Range**

YC_bC_r Color Model

peppers image

The Y component of peppers image

The C_b component of peppers image

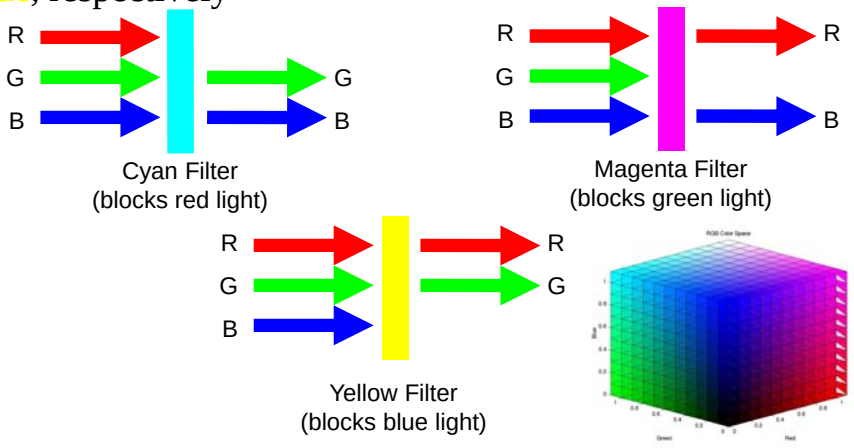
The C_r component of peppers image

CMYK Color Model

- The CMYK is a *subtractive* color system which frequently used by printers, normally on a white surface

CMYK Color Model

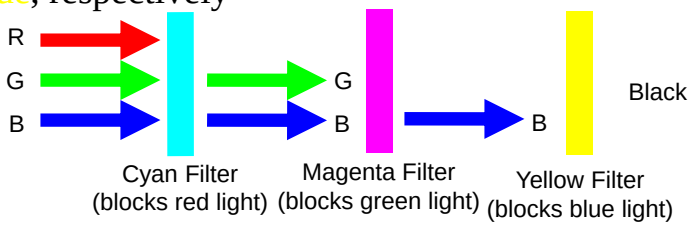
- When illuminated, cyan, magenta, and yellow filters absorb their complementary light color, which are red, green, and blue, respectively

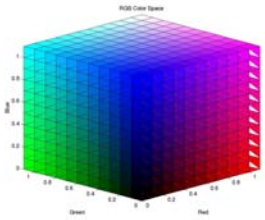


Topic 01: Digital Image Fundamentals

CMYK Color Model

- When illuminated, cyan, magenta, and yellow filters absorb their complementary light color, which are red, green, and blue, respectively

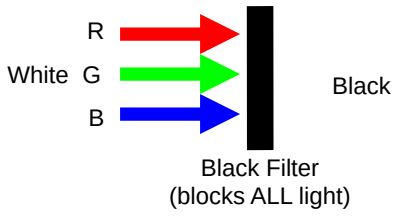




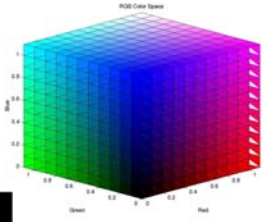
CMYK Color Model

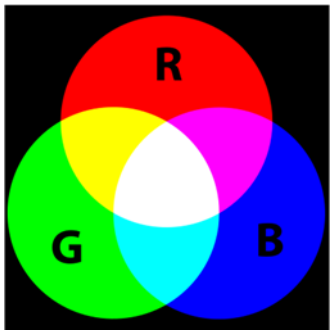
- In theory, when cyan, magenta, and yellow filters are combined, they absorb all light, hence resulting in black. However, printing black that way has major problems:
  - Colored ink is expensive; replacing colored inks by a black ink makes economic sense
  - Printing three ink layers causes the printed paper to become quite wet; replacing three inks by one will let the printed paper to dry more quickly
  - With mechanical tolerances, the three inks might be printed slightly out of register, hence edges will not be black
- To compensate all these problems, black color is tacked onto the color system and treated like an independent primary color variable; that is why CMYK scheme is often called 4-color printing

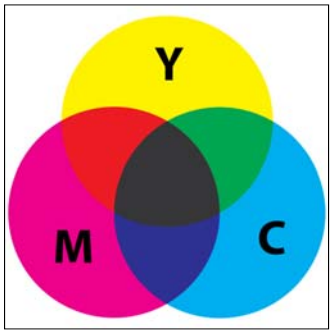
CMYK Color Model



CMYK Color Model







Additive color system

Subtractive color system

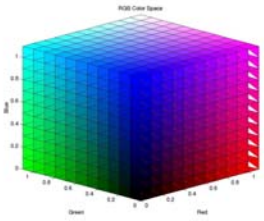
CMYK Color Model

- Conversion formula of RGB (ranged from 0 to 255) to CMY (ranged from 0 to 1)
  - The RGB values are divided by 255 to change the range from [0..255] to [0..1]

    $R' = R / 255$

    $G' = G / 255$

    $B' = B / 255$
  - $C = 1 - R'$
  - $M = 1 - G'$
  - $Y = 1 - B'$

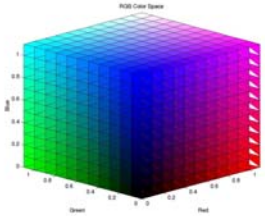


- Conversion formula of CMY (ranged from 0 to 1) to RGB (ranged from 0 to 255)
  - $R = (1 - C) \times 255$
  - $G = (1 - M) \times 255$
  - $B = (1 - Y) \times 255$

CMYK Color Model

- Conversion formula of RGB (ranged from 0 to 255) to CMYK (ranged from 0 to 1)
  - The RGB values are divided by 255 to change the range from [0..255] to [0..1]

    $R' = R / 255$
    $G' = G / 255$
    $B' = B / 255$
  - $C' = 1 - R'$
  - $M' = 1 - G'$
  - $Y' = 1 - B'$
  - $K = \text{Minimum}(C', M', Y')$
  - If K = 1
    THEN
    $C = M = Y = 0$
    ELSE
    $C = (C' - K) / (1 - K)$
    $M = (M' - K) / (1 - K)$
    $Y = (Y' - K) / (1 - K)$



- Conversion formula of CMYK (ranged from 0 to 1) to RGB (ranged from 0 to 255)
  - If K = 1
    THEN
    $R = G = B = 0$
    ELSE
    $R = 255 \times (1 - C) \times (1 - K)$
    $G = 255 \times (1 - M) \times (1 - K)$
    $B = 255 \times (1 - Y) \times (1 - K)$

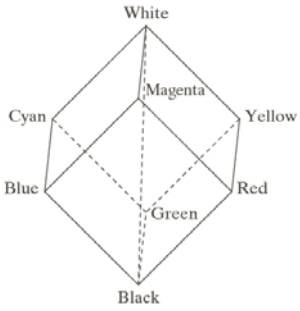
HSI Color Model

- Hue-Saturation-Intensity (HSI) is one of the most common ***cylindrical-coordinate*** representation of points in an RGB color model
- The ***hue*** (the color attribute that describes a pure color) is represented by the cylindrical angle
  - starting at the red at 0°,
  - passing through the yellow at 60°,
  - the green at 120°,
  - the cyan at 180°,
  - the blue at 240°,
  - the magenta at 300°, and
  - then wrapping back to red at 360°

- The ***saturation*** (a measure of the degree to which a pure color is *diluted by white*) is represented by the cylindrical radius (*inverse relation*); ranging ***from 0*** when having a pure gray shade (***at the centre***), ***to 1*** when having maximum color (***at the circumference***).
- The ***intensity***, the central vertical axis comprises the neutral gray colors, ranging from ***black at value 0*** at the bottom, to ***white at value 1*** at the top (*the above image has been generated at intensity value = 80%*)

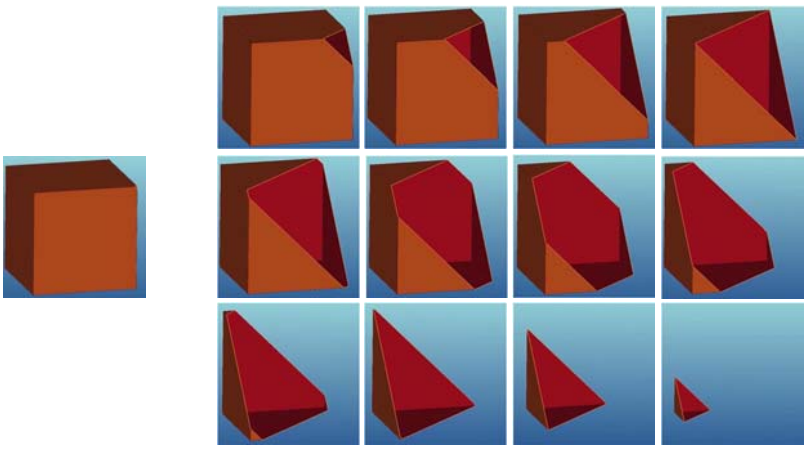
HSI Color Model

- The relationship between RGB and HSI models can be seen as:

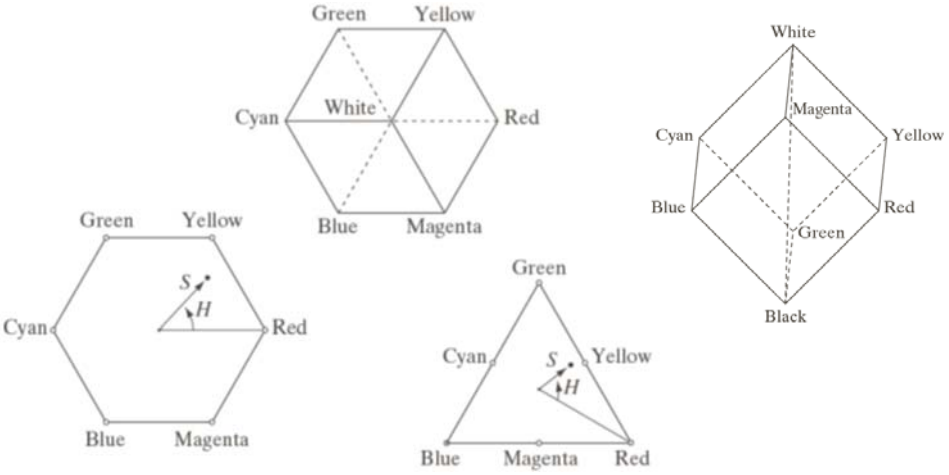


HSI Color Model

- What is the cube cross-sections across the main diagonal?

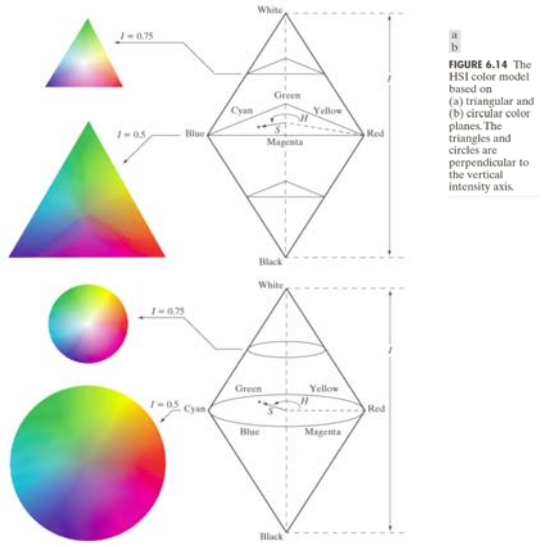


HSI Color Model



Hue and saturation in the HSI color model. The dot is an arbitrary color point. The angle from the red axis gives the hue, and the length of the vector is the saturation. The intensity of all colors in any of these planes is given by the position of the plane on the vertical intensity axis.

HSI Color Model



a
b

FIGURE 6.14 The HSI color model based on (a) triangular and (b) circular color planes. The triangles and circles are perpendicular to the vertical intensity axis.

HSI Color Model

- To convert from the RGB space to the HSI space, the following transformations may be used
- The RGB values are divided by 255 to change the range from [0..255] to [0..1]

$$\theta = \cos^{-1}\left\{\frac{(R-G)+(R-B)}{2\times\sqrt{(R-G)^2+(R-B)(G-B)}}\right\}$$

$$H = \begin{cases}\theta & if\ B \le G \\ 360-\theta & if\ B > G\end{cases}$$

$$I = \frac{(R+G+B)}{3}$$

$$S = 1 - \frac{3\times minmum(R,G,B)}{(R+G+B)} = 1 - \frac{minmum(R,G,B)}{I} = \frac{I - minmum(R,G,B)}{I}$$

HSI Color Model

To convert from the HSI space to the RGB space, the following transformations may be used	**If ($0° \le H < 120°$) i.e., RG sector** $B = I \times (1 - S)$ $R = I \times \left(1 + \frac{S \times \cos H}{\cos(60° - H)}\right)$ $G = 3 \times I - (R + B)$
If ($120° \le H < 240°$) i.e., GB sector $H = H - 120°$ $R = I \times (1 - S)$ $G = I \times \left(1 + \frac{S \times \cos H}{\cos(60° - H)}\right)$ $B = 3 \times I - (R + G)$	**If ($240° \le H \le 360°$) i.e., BR sector** $H = H - 240°$ $G = I \times (1 - S)$ $B = I \times \left(1 + \frac{S \times \cos H}{\cos(60° - H)}\right)$ $R = 3 \times I - (G + B)$

Color Versus Gray-scale

- Some display devices can not display colors at all, but rather shades of gray (gray-scale devices)
- Shade of gray can be represented by a single component
- In RGB model, the shades of gray occur when R = G = B

- To obtain a gray-scale version of a color image, you may use the G component from the RGB model, or for more accurate result, you may use the Y component from the YC_bC_r model

Color Versus Gray-scale


Peppers image


The Y component


Average R, G, and B


The R component

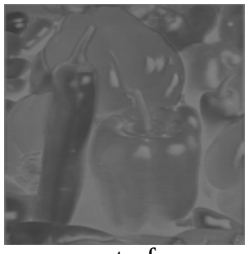

The G component


The B component

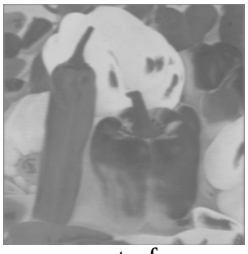
Color Versus Gray-scale

peppers image

The Y component of peppers image



The C_b component of peppers image



The C_r component of peppers image

True-Color Versus Color-Palette

- Any system which is capable of matching, or exceeding, the color-resolving power of the human eye under most conditions is called a true-color system
- In practice, true-color means 24 bpp
- Storing/transmitting a true-color image requires a huge amount of memory
- To overcome this problem, color-palette may be used
- A color-palette (a.k.a. color map, index map, color table, or look-up table) is a one-dimensional array of 3-byte elements that specify the color
- Using a color-palette, data in a file can be stored as a series of color index values, rather than directly specifying the true-color
- The most common size for a color-palette is 256 entries; i.e., each pixel index value consists of 8 bits
- ***How are these color-palettes defined?***
- ***What will we get if we display these index values without converting them into their true-colors?***

Byte Ordering And Bit Ordering

- All bitmap image files contain integers stored in a binary format
- For *single-byte* integers (i.e., characters), there is *no compatibility* problem among different *processor types*
- However, when reading *multi-byte integers*, there is an issue of *how to order bytes* from the input stream into the integer number, i.e., does the *most significant byte occur first, or last, in the file*?
  - If the most significant integer byte is stored first, we call it *big-endian*; (which means big, or most significant byte, stored first)
    Example: Motorola processors (used in SUN SPARC family)
  - On the other hand, if the least significant integer byte is stored first, we call it *little-endian*; (which means little, or least significant byte, stored first)
    Example: Intel processors (used in most PCs)

Byte Ordering And Bit Ordering

- Consider that you stored a file called "text.txt" on a disk, which is connected to a SUN machine and a Linux PC
- When you execute **"type text.txt"** from the SUN machine or the Linux PC, you get:
  abcde
- However, when you execute **"od -cx text.txt"** from:
  - The SUN machine, you will get:

```
0000000   a   b   c   d   e  \n
              6162    6364    650a
0000006
```

  - The linux PC, you will get:

```
0000000   a   b   c   d   e  \n
              6261    6463    0a65
0000006
```

Byte Ordering And Bit Ordering

```
int main(int argc, char * argv[])
{  FILE  *dest_fp;
   short int my_variable=64;

   if (argc != 2)
   { fprintf(stderr,
        "usage: %s dest_file\n", argv[0]);
     exit(EXIT_FAILURE);
   }
   if(!(dest_fp = fopen(argv[1], "w")))
   { fprintf(stderr,
        "%s can not be opened\n", argv[1]);
     exit(EXIT_FAILURE);
   }
   fwrite(&my_variable,
            sizeof(short int), 1, dest_fp);

   fclose(dest_fp);
   return 0;
}
```

```
int main(int argc, char * argv[])
{  FILE *source_fp;
   short int my_variable=0;

   if (argc != 2)
   { fprintf(stderr,
        "usage: %s source \n", argv[0]);
     exit(EXIT_FAILURE);
   }
   if (!(source_fp = fopen(argv[1], "r")))
   { fprintf(stderr,
        "%s can not be opened\n", argv[1]);
     exit(EXIT_FAILURE);
   }
   fread(&my_variable,
            sizeof(short int), 1, source_fp);
   printf("%d\n", my_variable);
   fclose(source_fp);
   return 0;
}
```


Byte Ordering And Bit Ordering

- If you compile these two programs and run them on the same computer, you will get the following:

```
Linux_Machine[11]% dest_prog aaa
Linux_Machine [12]% source_prog aaa
64
Linux_Machine [13]%
```

Or

```
SUN_Machine[11]% dest_prog bbb
SUN_Machine[12]% source_prog bbb
64
SUN_Machine[13]%
```

00000000	01000000

The binary representation of **my_variable**

What are the internal representations of `aaa` ***and*** `bbb` ***files?***

Byte Ordering And Bit Ordering

- If you send the file aaa to the SUN_Machine and run the program again, you will get:

```
SUN_Machine[13]% source_prog aaa
16384
SUN_Machine[14]%
```

- If you send the file bbb to the Linux_Machine and run the program again, you will get:

```
Linux_Machine [13]% source_prog bbb
16384
Linux_Machine [14]%
```