ECE5554 – Computer Vision
Lecture 1b – Images in Memory

Creed Jones, PhD

BRADLEY DEPARTMENT
OF ELECTRICAL & COMPUTER ENGINEERING

VIRGINIA TECH.

Today's Objectives

- An Image in Memory
- Color Image Sensing
  - Bayer Filters and Demosaicing
- A Color Image in Memory
- Common Color spaces
  - Red – Green – Blue (RGB)
  - Hue – Saturation – Intensity (HSI)
  - RGB ←→ HSI Conversion
- Other Color Spaces
- Image Quality Metrics

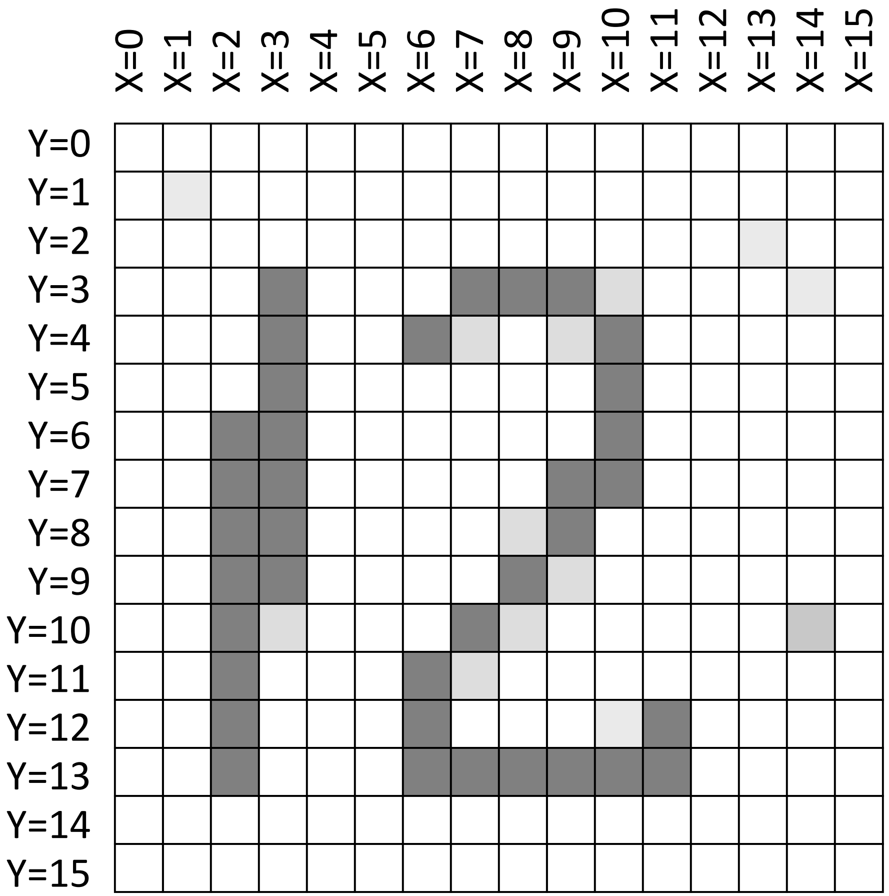
AN IMAGE IN MEMORY

Image Processing is the use of mathematically based operations, via computer, to obtain a useful result from a digitally stored image

So, what's an image?

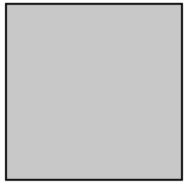
- A digital image is a collection of image values representing a real-world scene
- A digital image is a matrix of brightness or color values (matrix? brightness?)
- A digital image is a two-dimensional function *f(x,y)* where *x* and *y* are spatial coordinates, and f() is <u>some measure of appearance</u>

A digitally stored image is made up of picture elements called *pixels*



- Each pixel is a single intensity value (light to dark) or maybe color
- Uniform rectangular grid
- X is horizontal
- Y is vertical
- (x,y) = (0,0) is in the upper left corner (usually)

A pixel



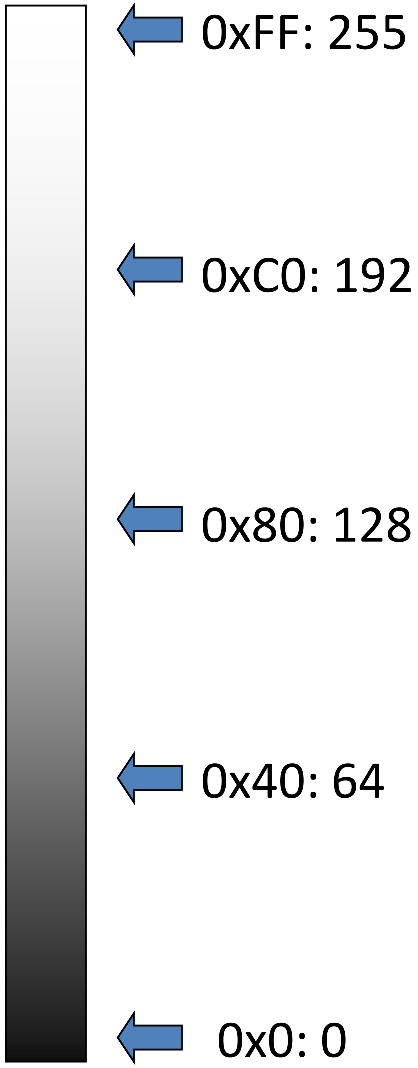
`Image[y0][x0]`

$$f\left(x,y\right)\Big|_{\substack{x=x_0\\ y=y_0}}$$

- Picture element
- Indivisible (we can’t distinguish image variations smaller than one pixel – not <u>directly</u>, anyway)
- Has a location in the image (is *spatial*)
- Has an intensity value – light to dark
- Often this is from 0 (blackest black) to 255 (whitest white)

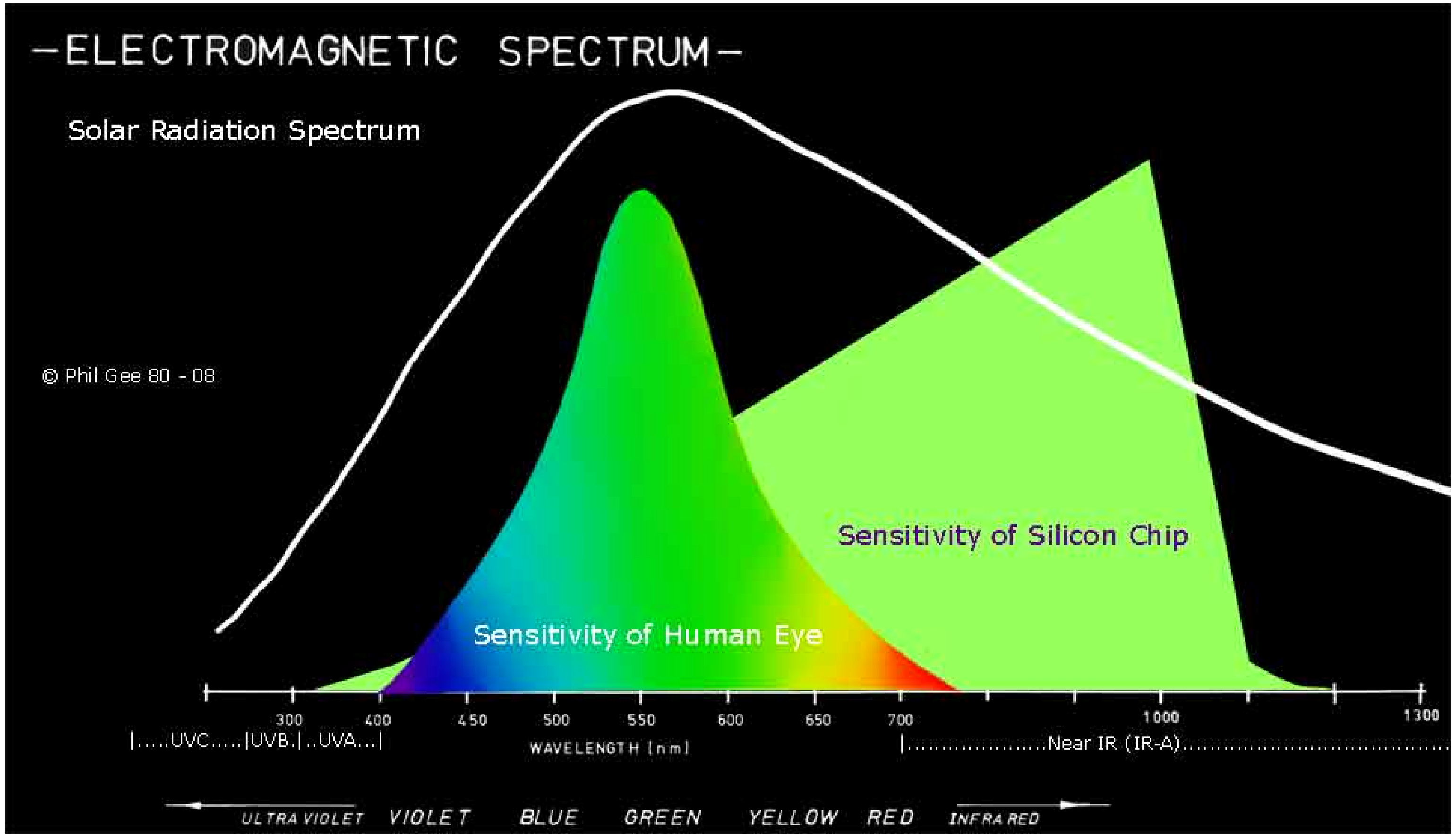
The image intensity is represented by *gray-scale* values

- Relation between image intensity and the pixel's numeric value
- Most common is "positive-bright", with lighter area having higher values
- This one is 8-bit gray scale
  - there are other options
- This one is *linear* – we will discuss later



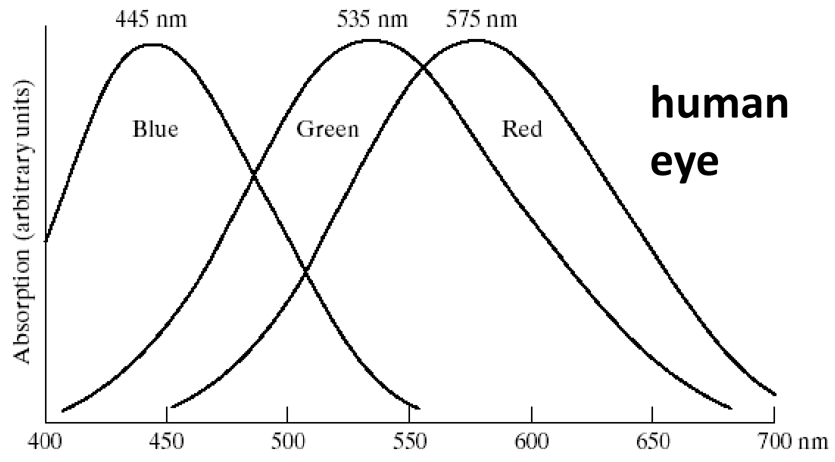
COLOR IMAGE SENSING

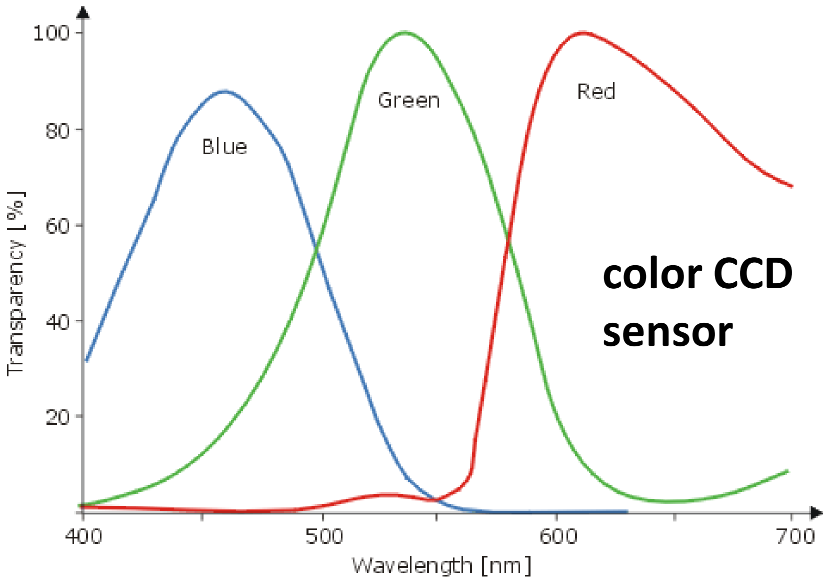
Image sensors are built on silicon wafers, which have a higher response in the infrared than human visual cells



http://philipgee.com/ElecSpec-01.html

The spectral sensitivity of image sensors is similar to that of the eye – but there are differences





- Note that the blue and green cones have responses very similar to a color camera
- Visual "red", however, is much more yellow than camera red
  - We don't really do anything about this
- Note also the camera response's tail extending into the IR
  - Many cameras include a glass IR cut filter to make the red response more like the visual system

A single CMOS camera pixel contains the same image capture features as a monochrome image sensor

- Recall that each pixel can only produce one intensity reading
- This one has a red optical filter, so it will sense the amount of red light in the image
- We arrange pixels sensitive to red, green and blue on the sensor, and produce three images
- but the red, green and blue pixels aren’t lined up!!!! (well, close enough)

Anatomy of the Active Pixel Sensor Photodiode

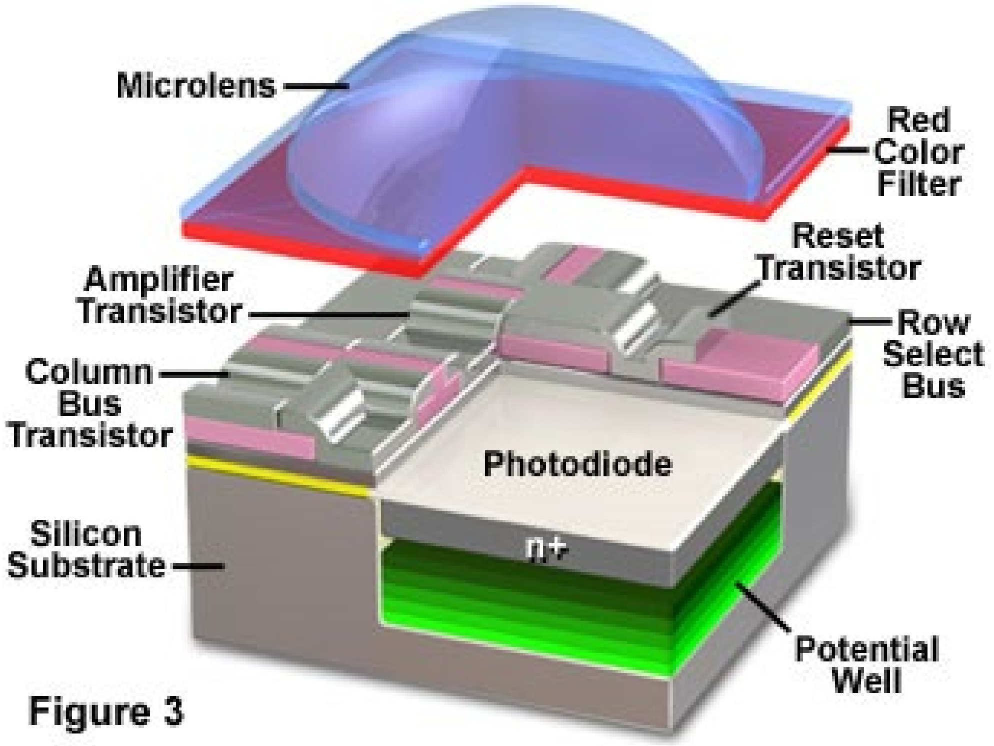
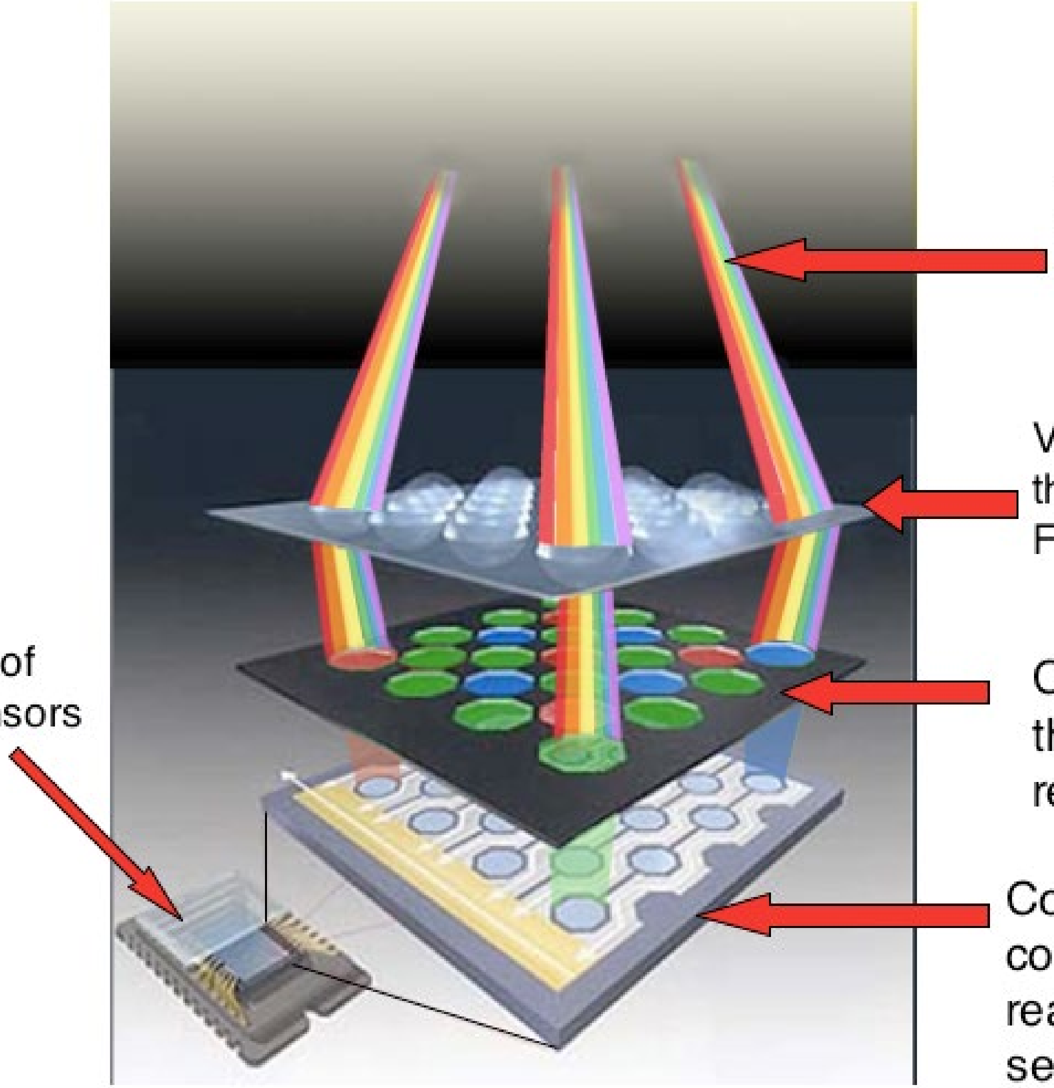


Figure 3

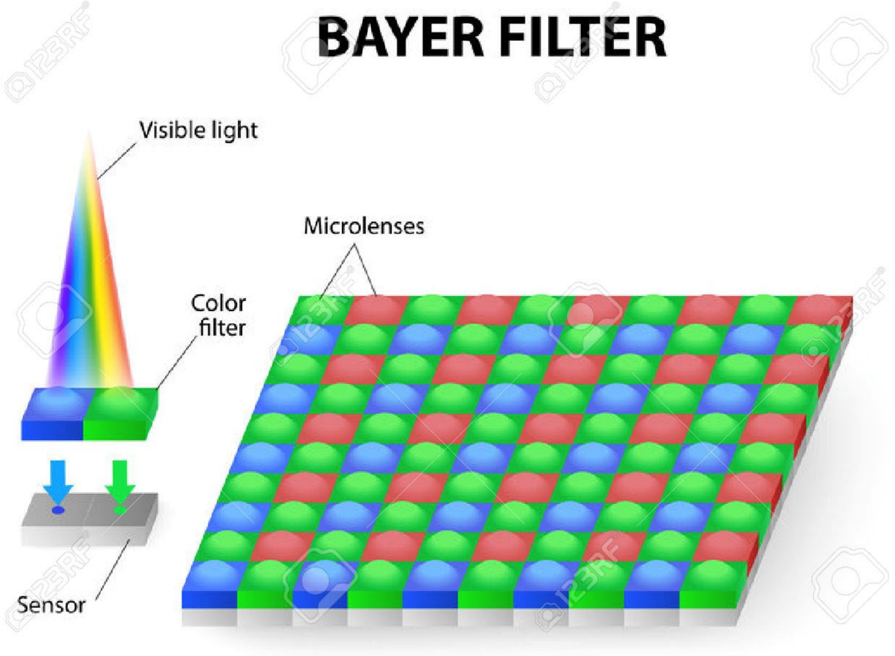
RGB Inside the Camera



http://www.globalspec.com/learnmore/video_imaging_equipment/video_cameras_accessories/cmos_cameras

The Bayer Filter is a mosaic of optical filters and microlenses that give each pixel a sensitivity to a specific color range

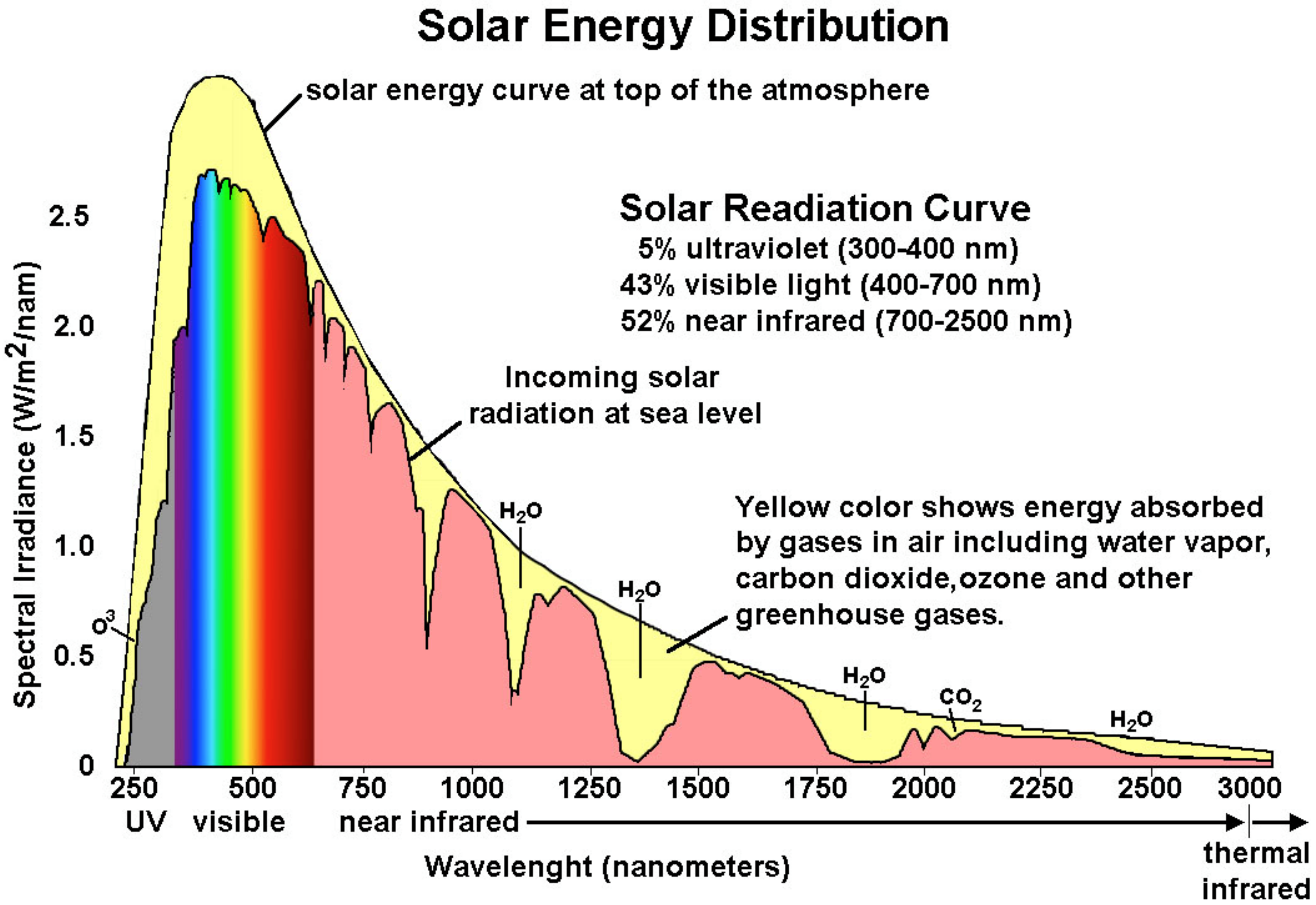
- US Patent #3971065
- The raw image is therefore composed of pixels that are only sensitive to one color
- To produce an image where each pixel location has red, green and blue components, a *demosaicing* algorithm is applied – usually in the camera itself



Demosaicing algorithms estimate the color information for certain pixels from the neighborhood

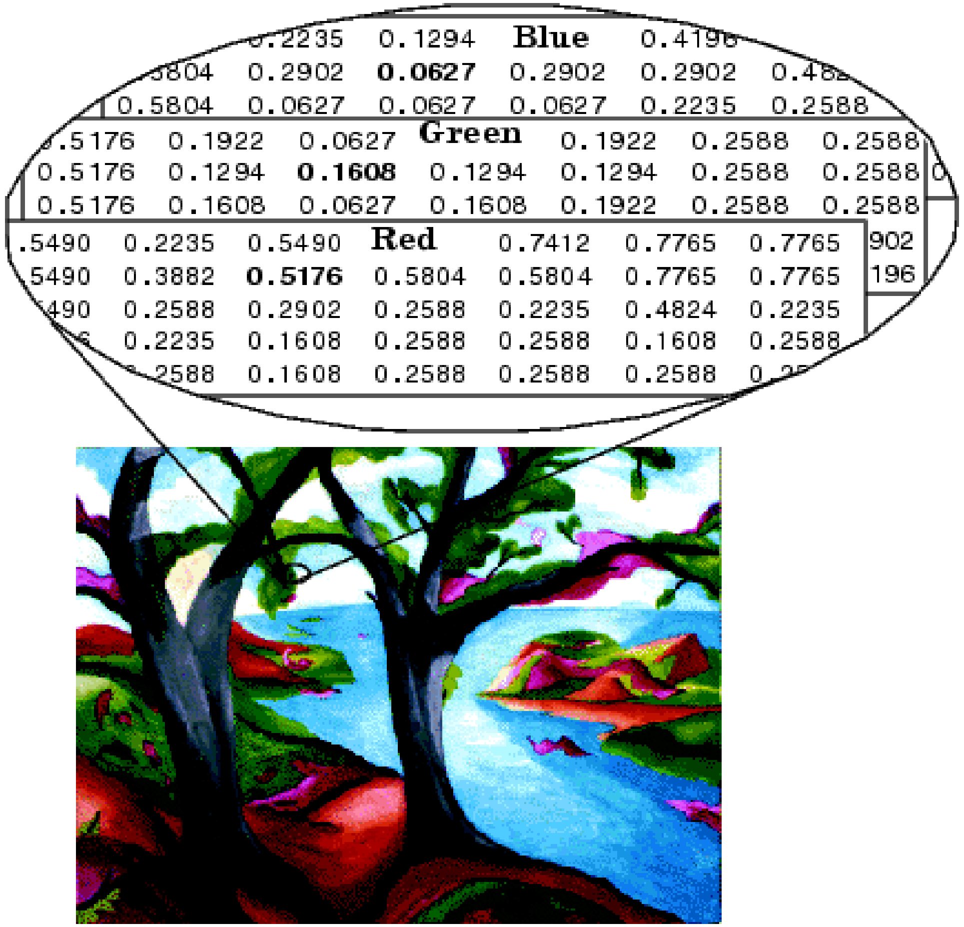
- The problem: for a pixel location that was under a red filter, we have the red value, but what were the blue and green values?
- Algorithms can be simple:
  - for each missing value, interpolate the surrounding known values
- If more processing is possible, more sophisticated methods can be used
  - estimate based on surrounding values and the gradients
  - interpolate in a manner designed to preserve local smoothness and homogeneity
  - "pixel grouping" methods use gradients as well
- This article describes these and some more advanced methods: http://www.photo-lovers.org/pdf/gunturk05.pdf

The illumination that we are using also has spectral characteristics – even sunlight has a complex spectral curve



A COLOR IMAGE IN MEMORY

The data from these sensors forms a color image – each pixel contains three pieces of information

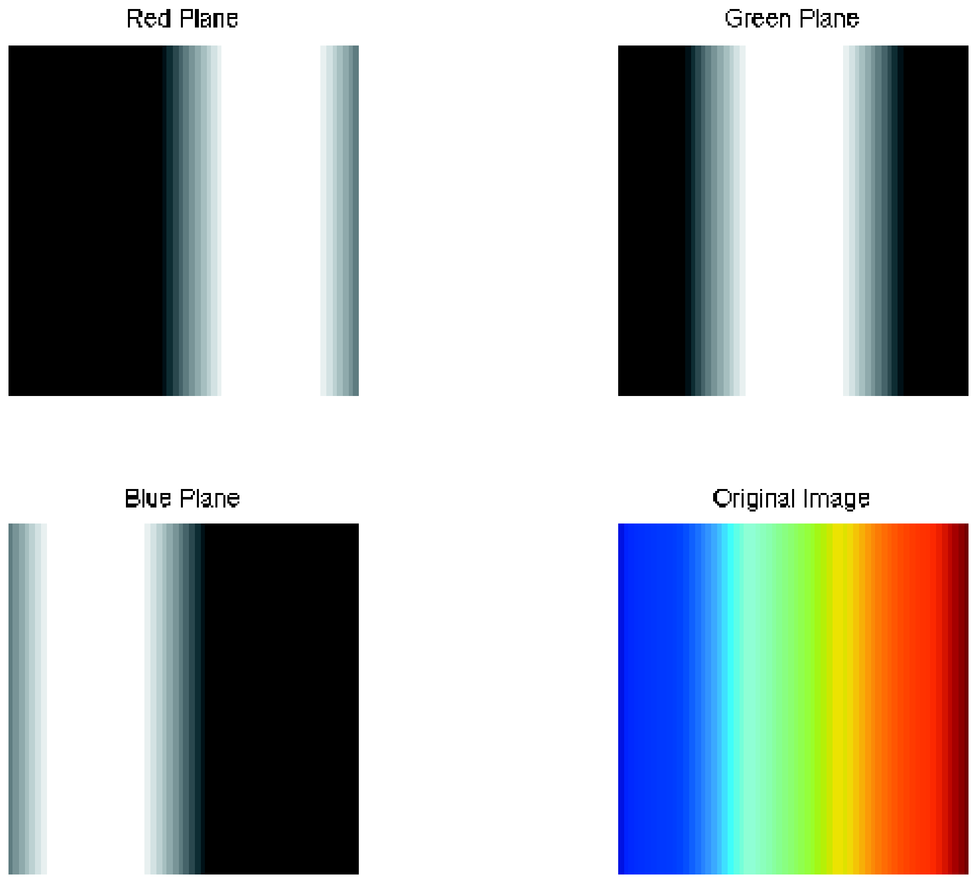


- Vector-valued image
  - or *tensor*
- Three co-located images
  - each one represents the intensity in its color band

$$Img(x,y) = \begin{bmatrix} p_{red} \\ p_{green} \\ p_{blue} \end{bmatrix}$$

A color image in three components

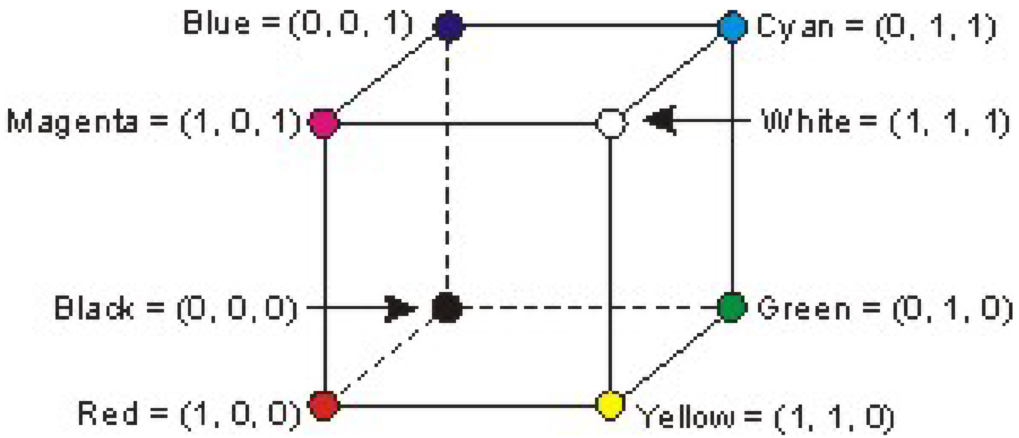
- Each component image is bright where the color image is high in that color



There are many choices for representing a color image in a matrix

- Nearly all use three components
- Often think of color space as three-dimensional
- Some are more natural for capture and display
- Others are more natural for some kinds of processing

RGB color space can be thought of as 3D Cartesian space



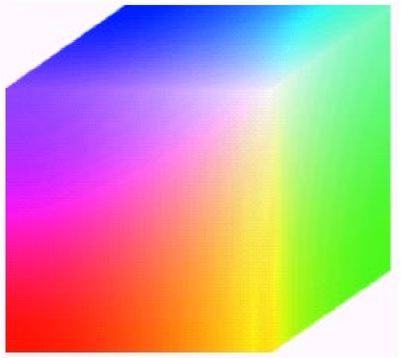


FIGURE 6.8 RGB 24-bit color cube.

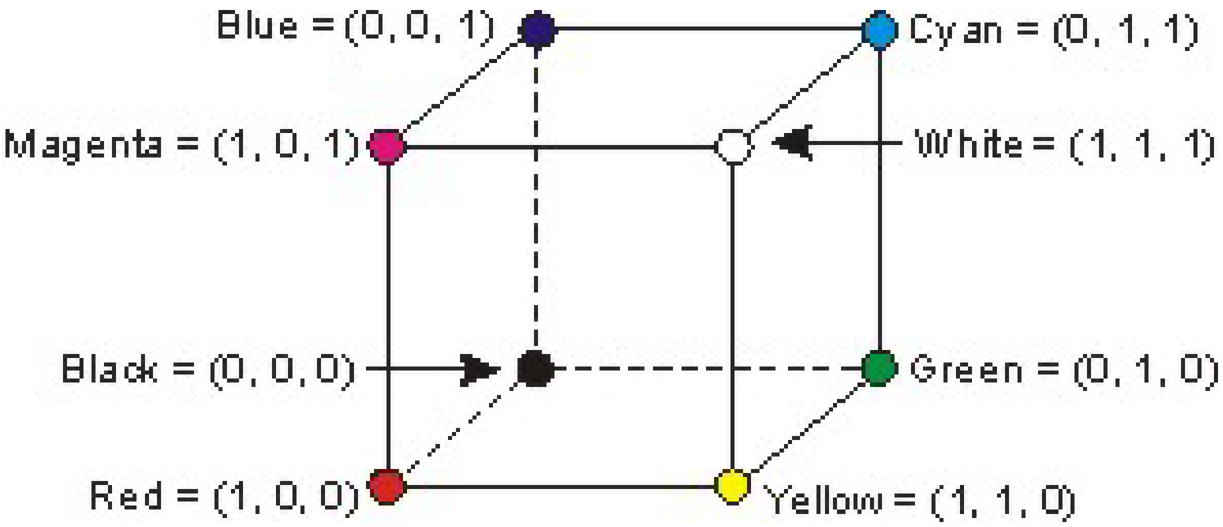
- Red, Green and Blue are the 3 axes
- Black is at the origin
- Shades of gray are along the diagonal of the cube
- Widely used in color cameras and displays

RGB is the most common color space used for representation

- It displays nicely
- It is a lot like human vision
- In an 8-bit system:
  - White objects have R, G and B = 255
  - Black objects have R, G and B = 0
  - The most vivid blue has R=G=0 and B = 255
- Other colors are obtained by other variations
  - you get the idea…
- Associate each color with a point somewhere inside the cube…

Color variations in RGB space

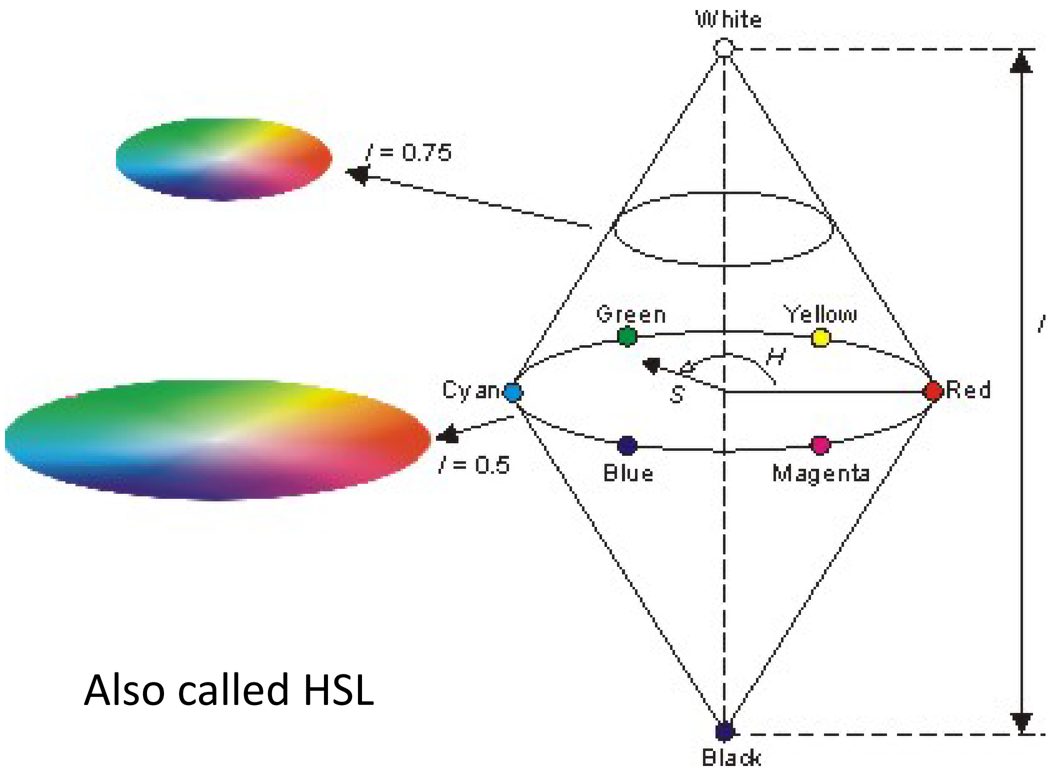
- Grays are found along the diagonal line where R=G=B (no discernible tone)
- To make a color darker without changing the tone, move along the line towards the origin
  - To make lighter, move out along the same line
- Moving towards any of the corners changes the tone



RGB has limitations

- It is not easy to understand how some common colors are represented
  - Where is brown in the RGB cube?
- Some very vivid colors are not representable, because they are outside the cube
  - A little hard to understand
- Color “tone” is not easy to derive from the RGB representation
- It is often handy to have a single plane that represents the “color” of objects
  - all the reds are together, etc.

HSI color space is a three-dimensional representation where the components are *Hue, Saturation* and *Intensity*



Also called HSL

- Hue – color “tone”
  - what color is it?
- Saturation – color intensity
  - how vivid is it?
- Intensity – brightness
  - how light is it?

These RGB / HSI conversions are used to convert the color of any pixel from one representation to another

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

Saturation $S = 1 - \frac{3(\min(R,G,B))}{R+G+B}$

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

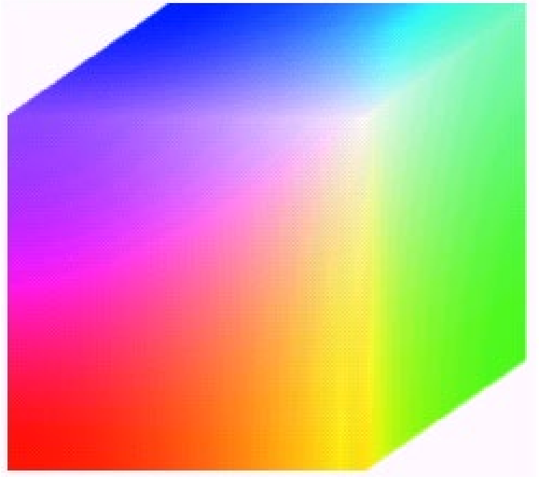
Red $R = I\left[1 + \frac{S\cos(H)}{\cos(\pi/3 - H)}\right]$

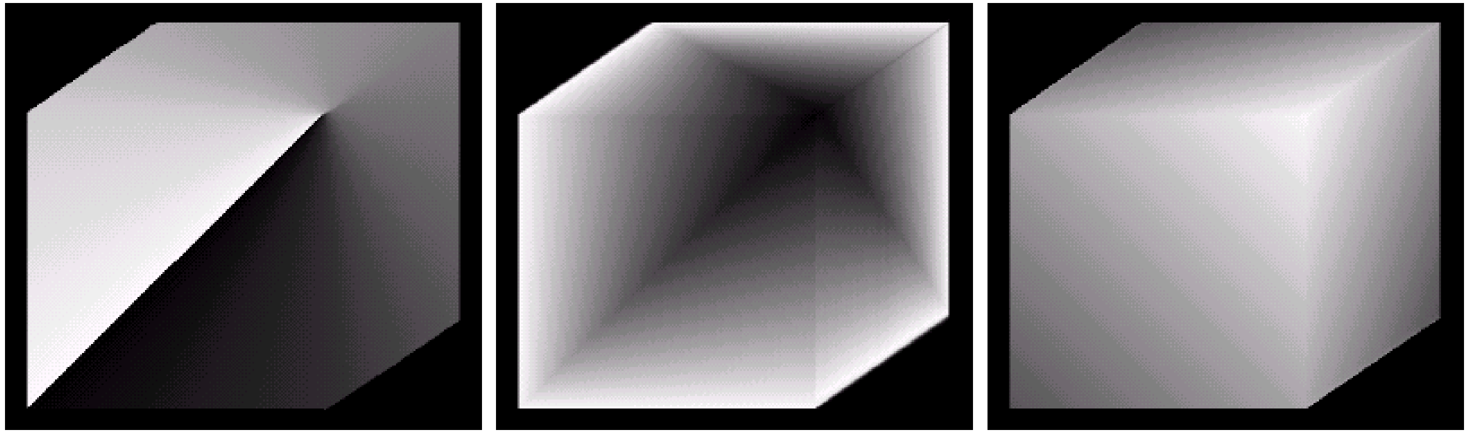
Blue $B = I(1-S)$

Green $G = I\left(S + 1 - \frac{S\cos(H)}{\cos(\pi/3 - H)}\right)$

- Before using these conversions, normalize the color values to the range $[0.1]$
- The conversions are nonlinear
- There are slightly different versions of these formulae
- Notice that saturation will go up as any color goes down with respect to the others

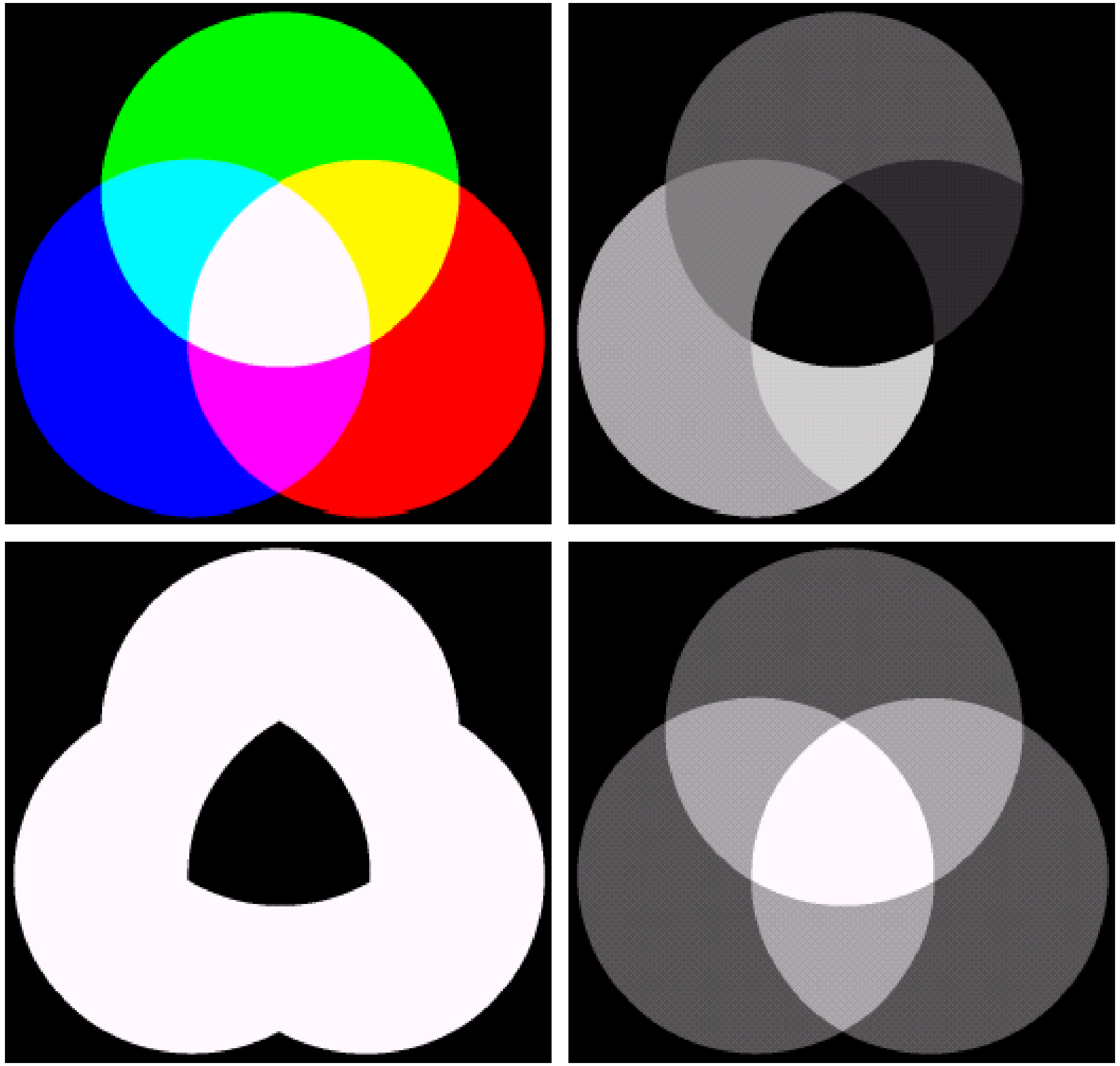
Relating colors to HSI





a b c

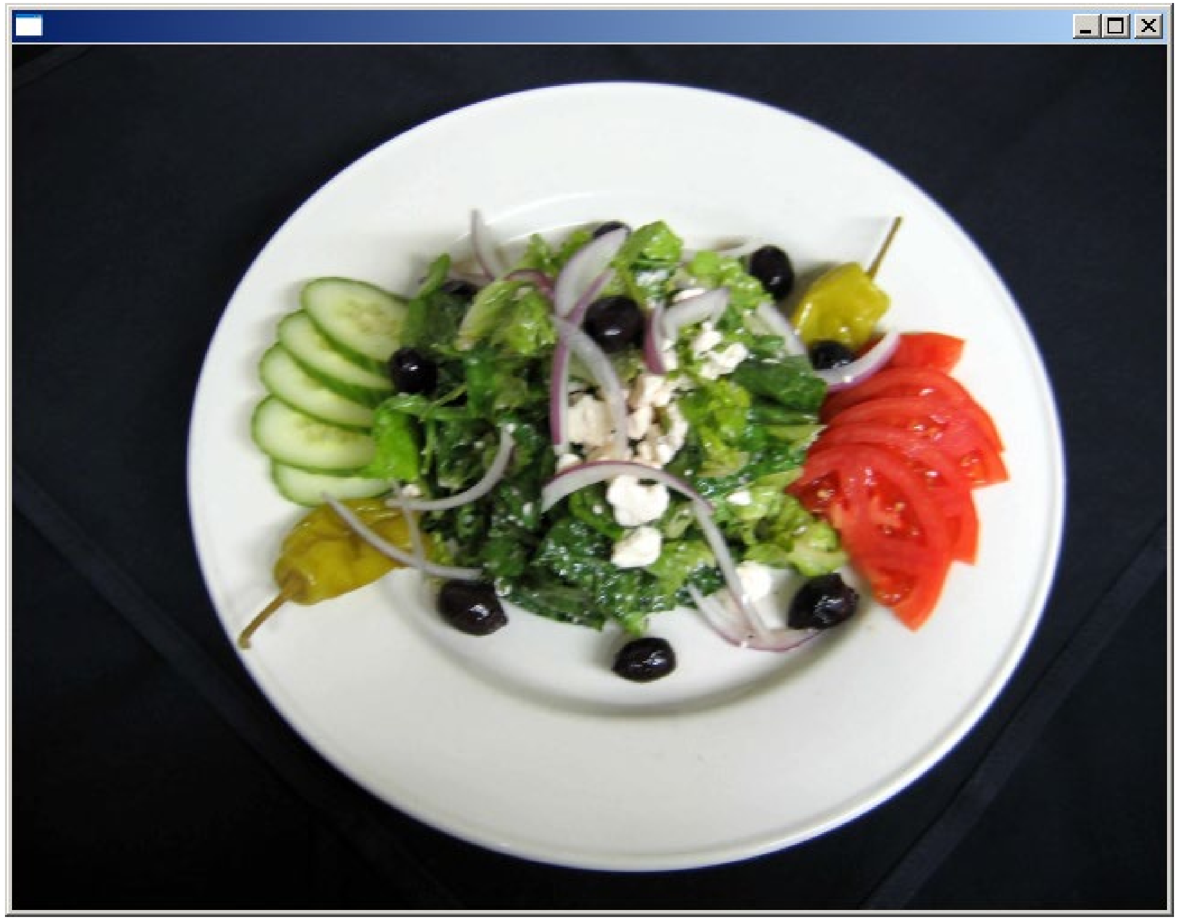
FIGURE 6.15 HSI components of the image in Fig. 6.8. (a) Hue, (b) saturation, and (c) intensity images.

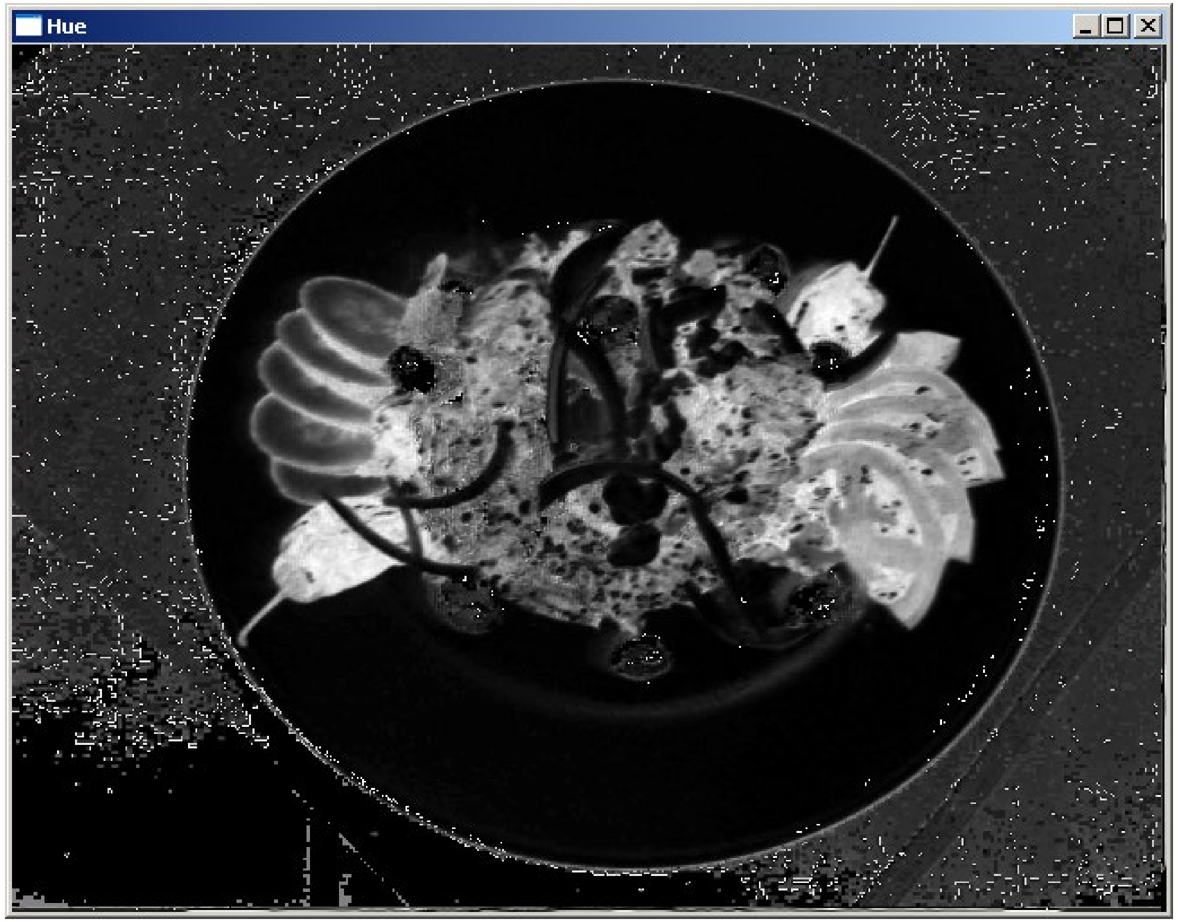


a	b
c	d

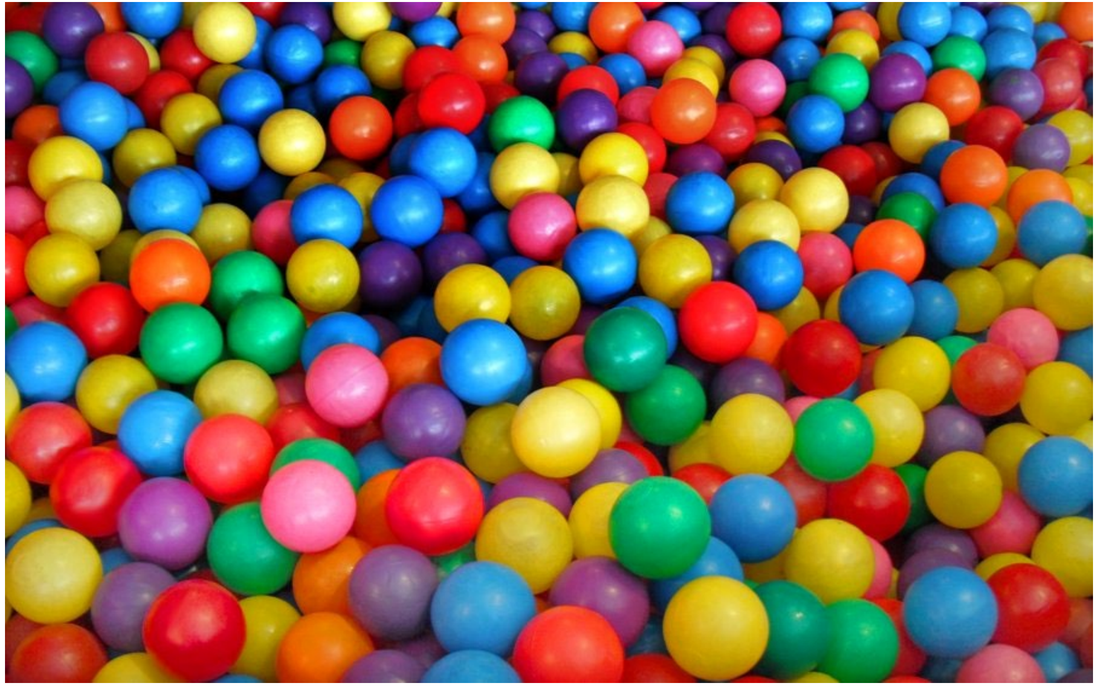
FIGURE 6.16 (a) RGB image and the components of its corresponding HSI image: (b) hue, (c) saturation, and (d) intensity.

Hue images indicate color for pixels with non-trivial saturation values (what color is gray? white? mathematically unstable quantities can result)

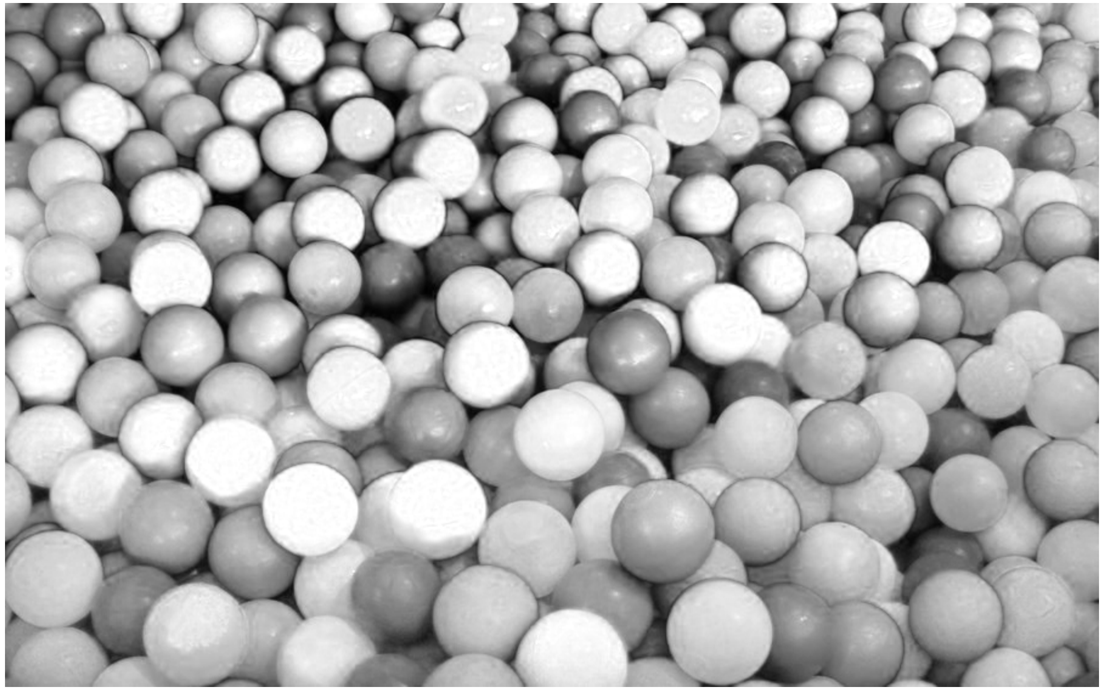




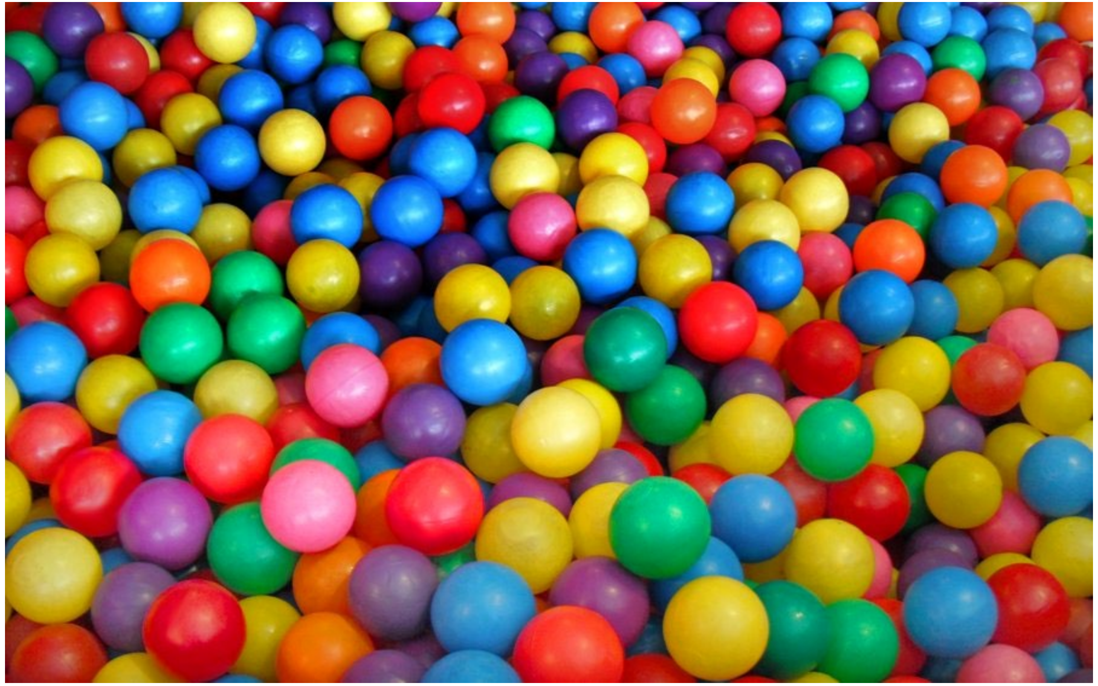
Color Image



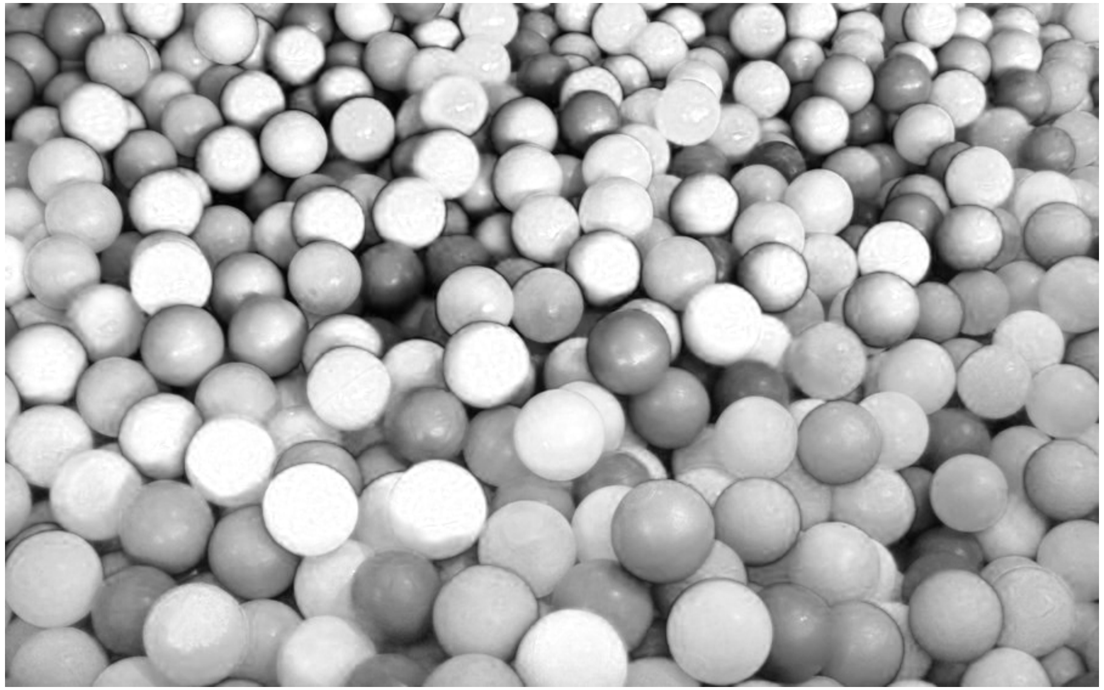
Intensity Image

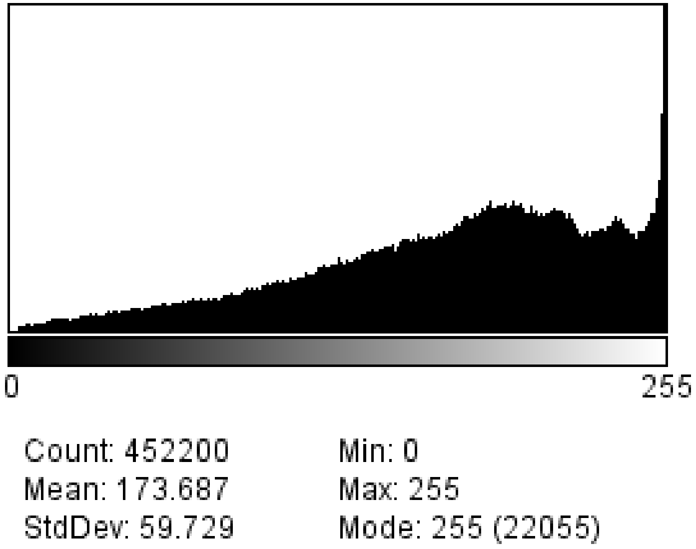


Color Image

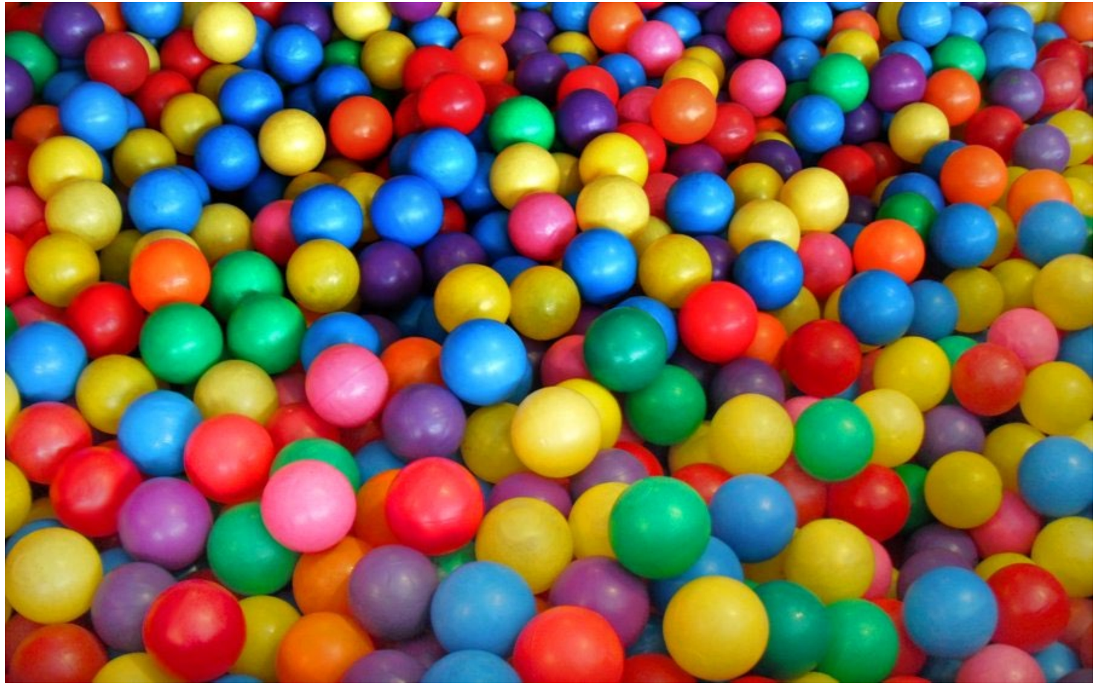


Intensity Image

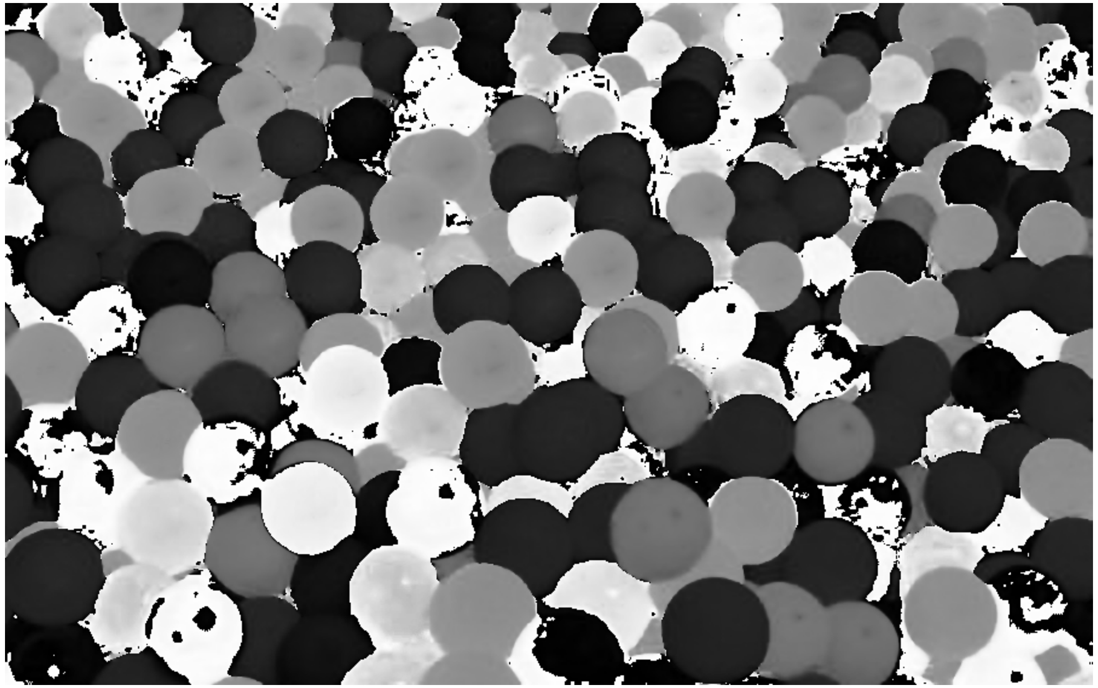




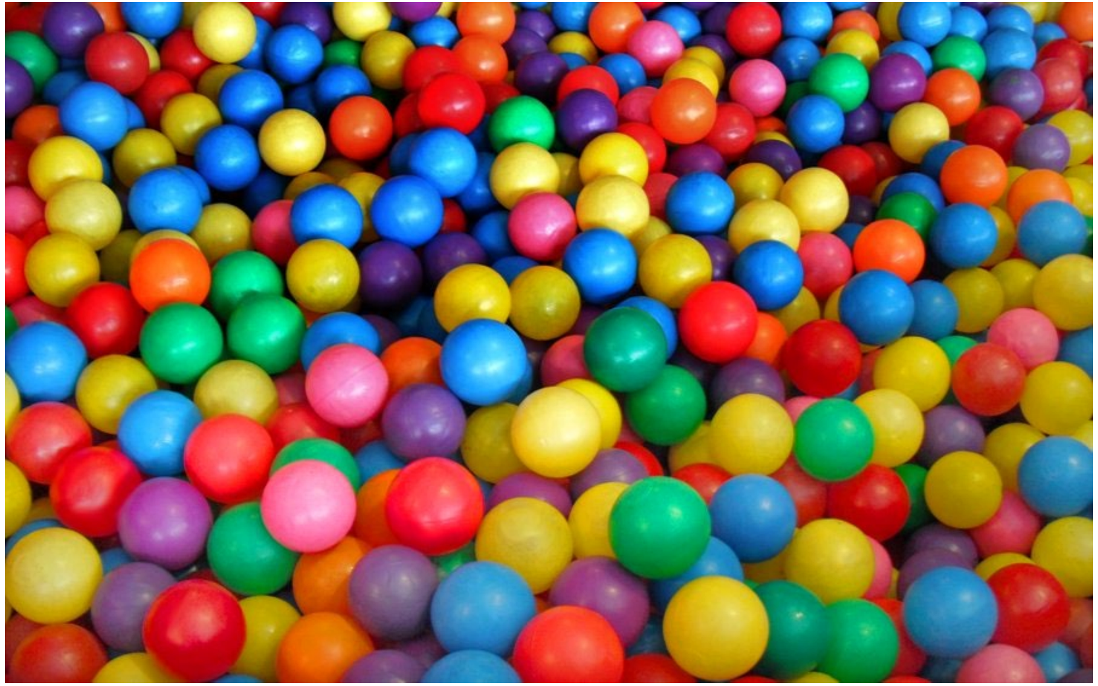
Color Image



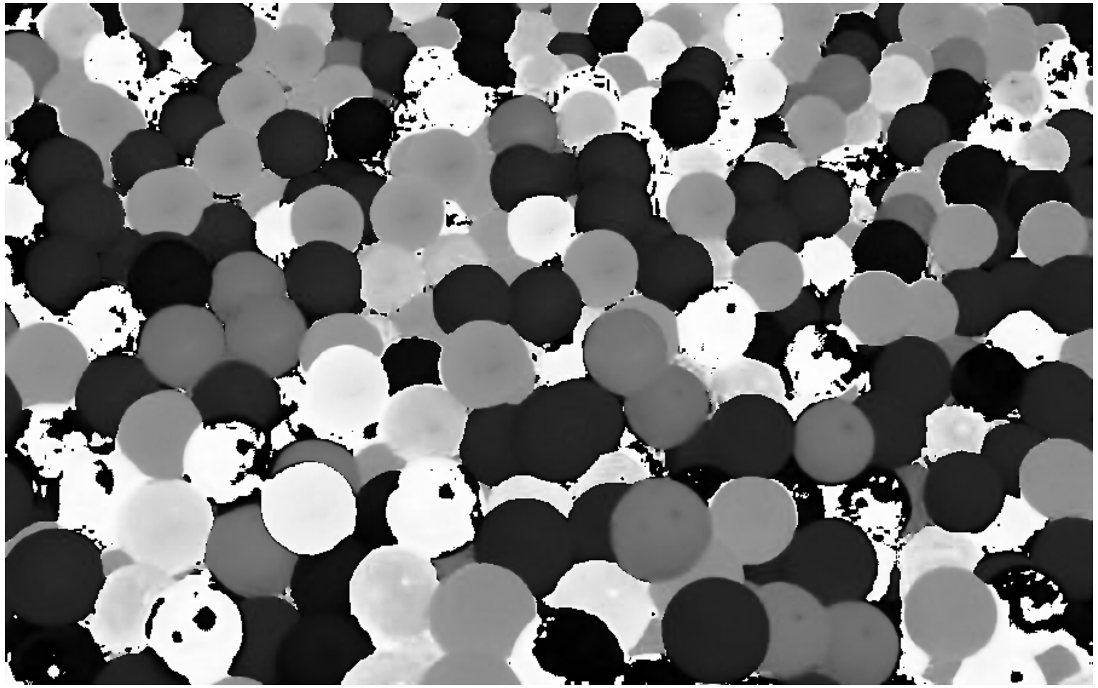
Hue Image

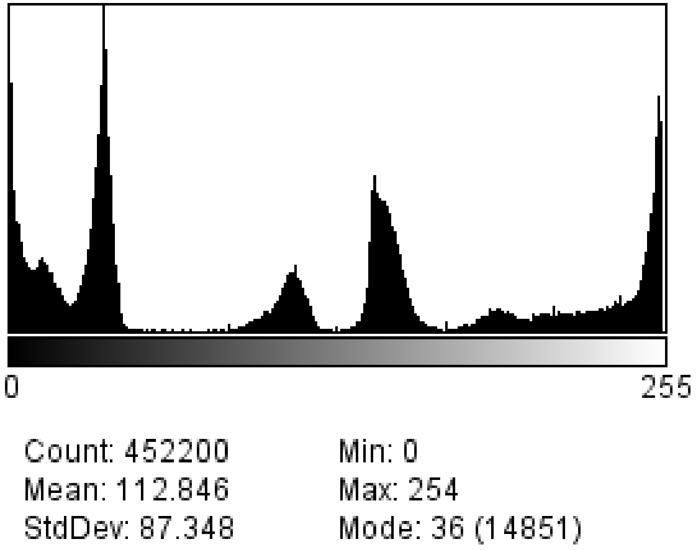


Color Image

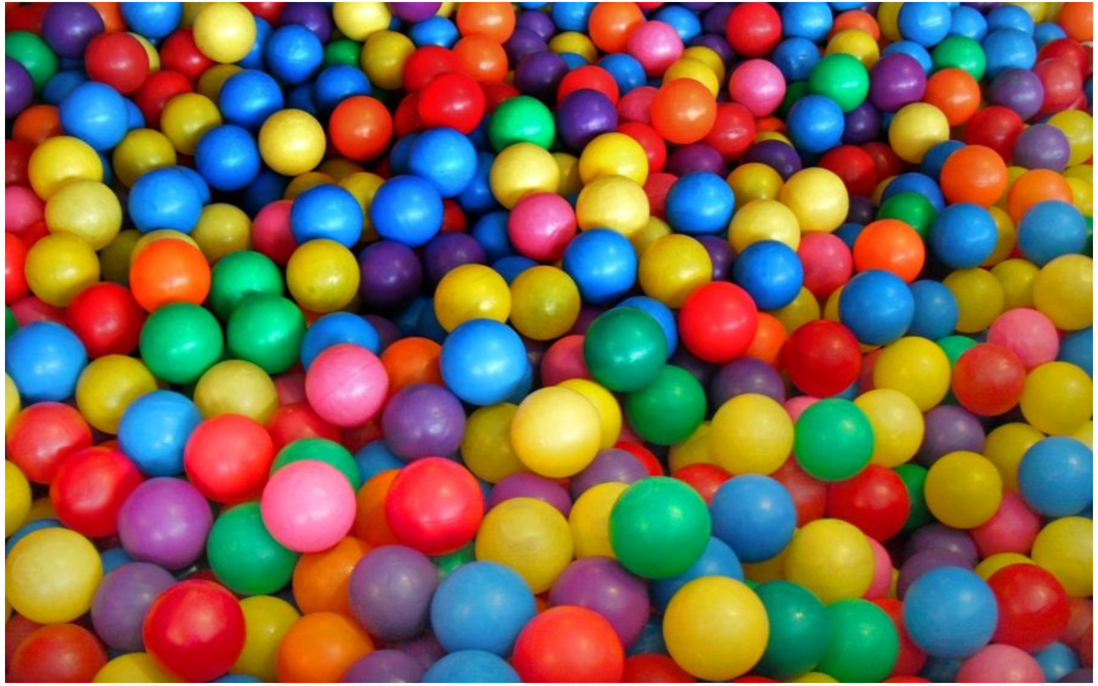


Hue Image

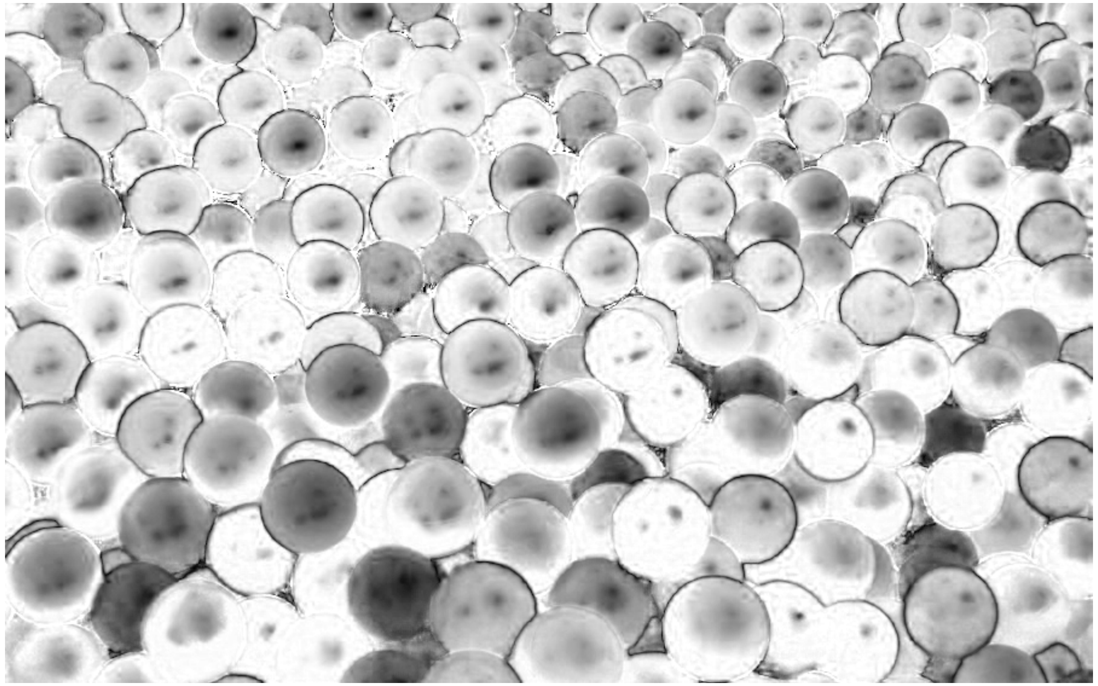




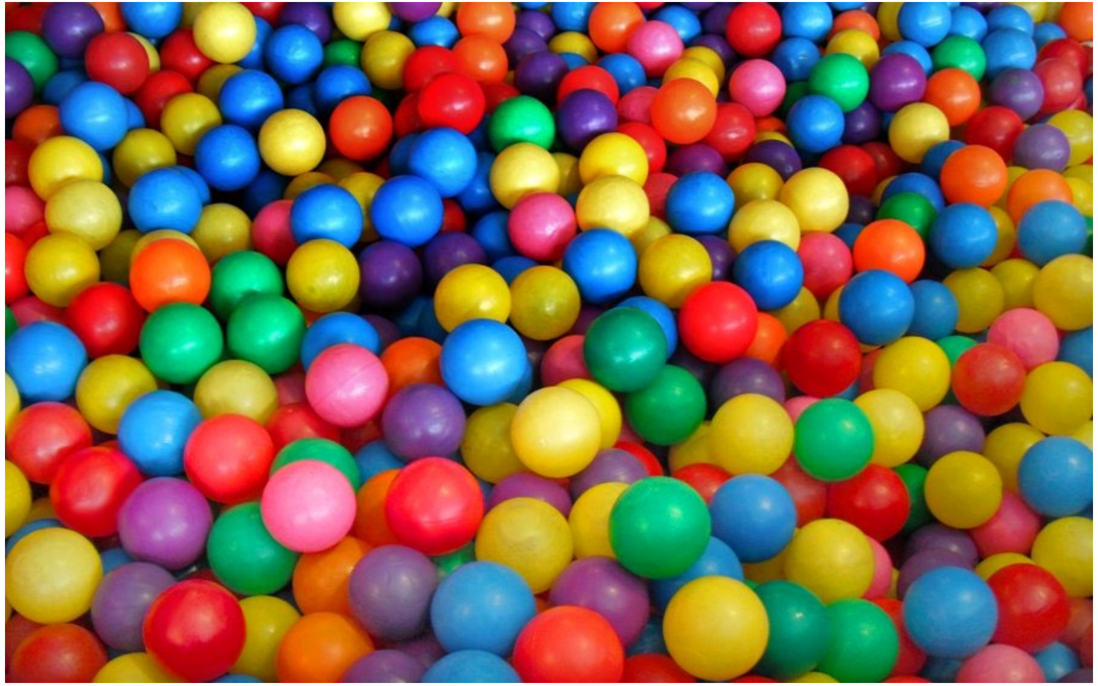
Color Image



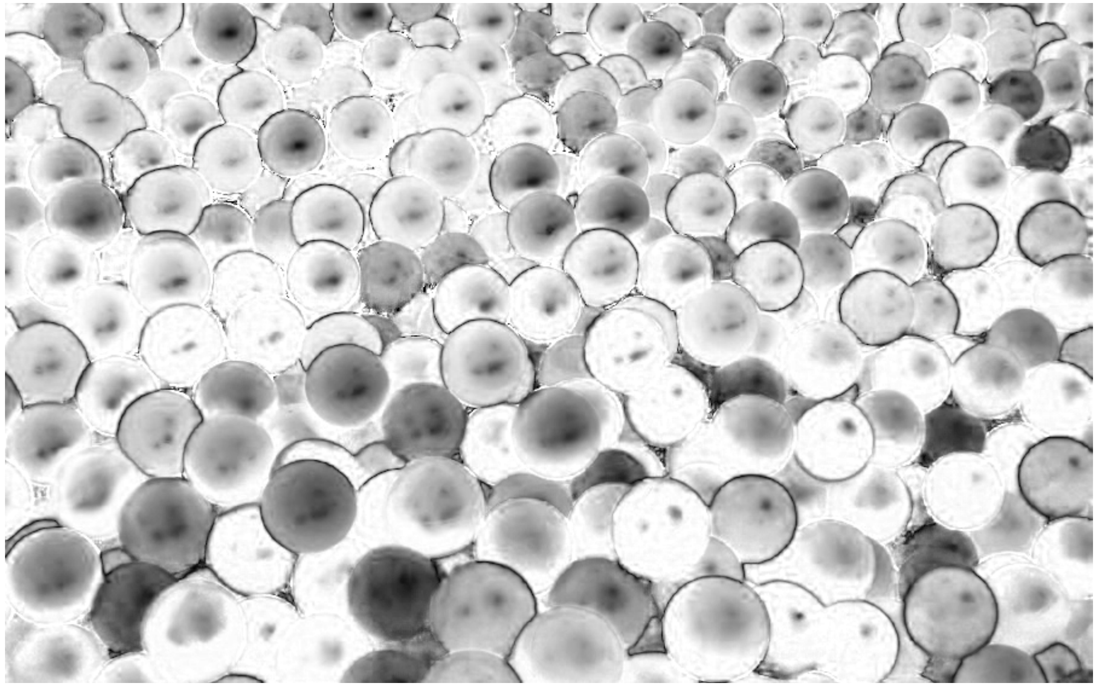
Saturation Image

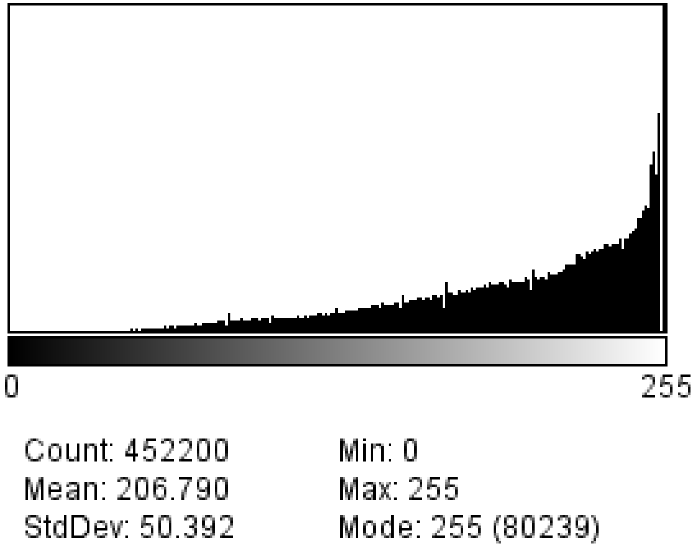


Color Image

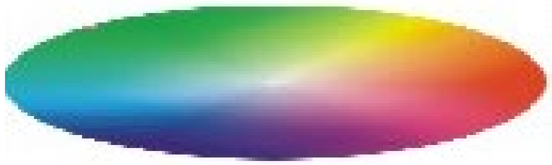


Saturation Image

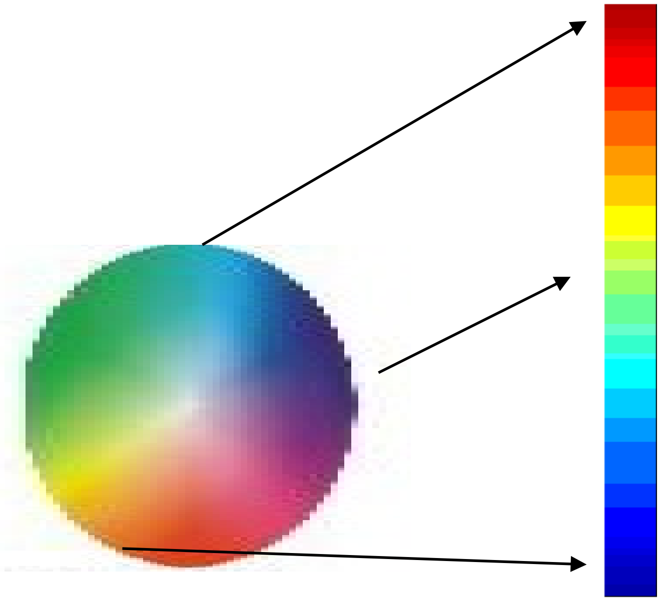




Hue is a circular quantity

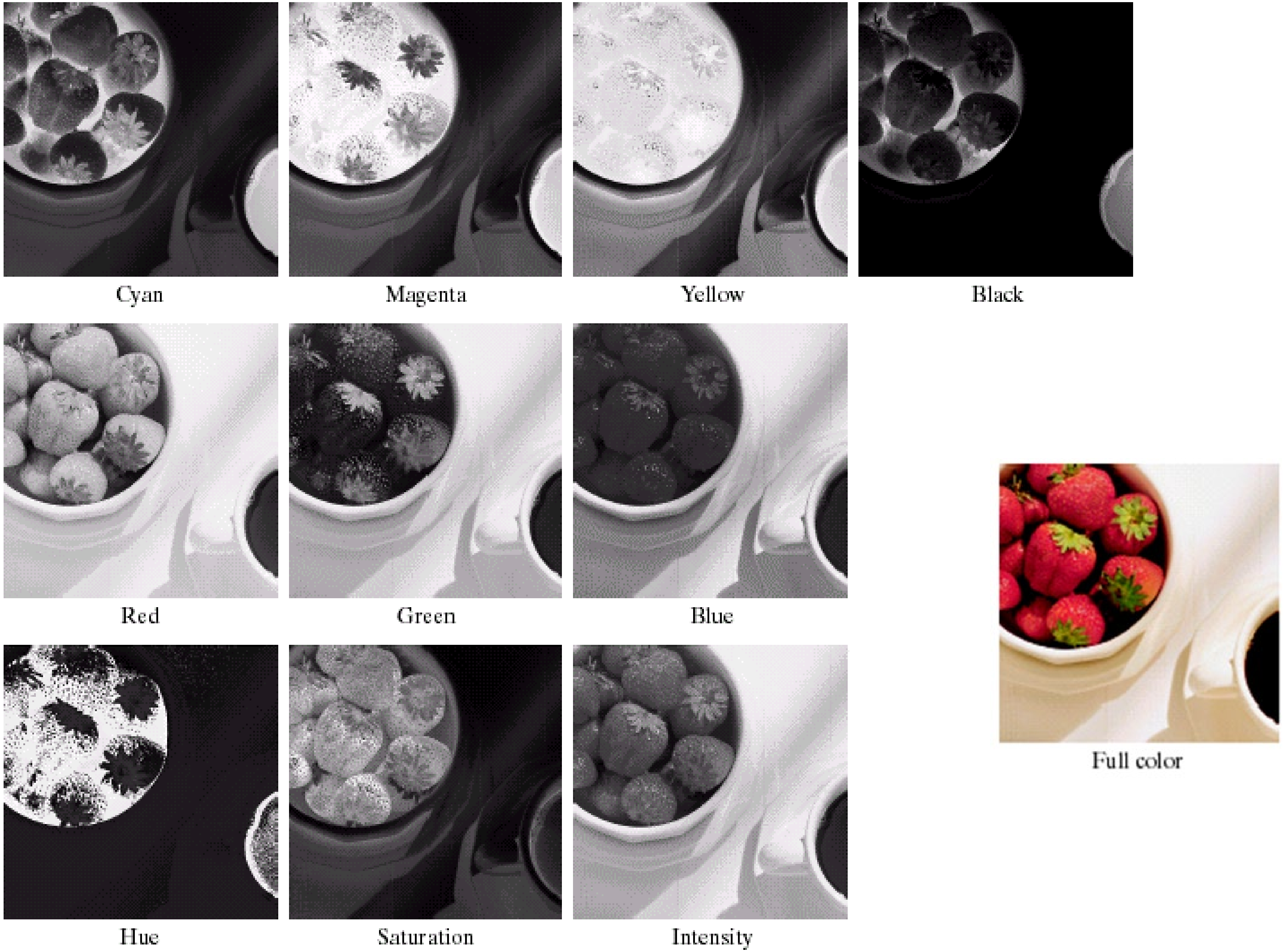


- Notice that hue is represented as an angle
  - Position around the color wheel…
- This seems OK, but think about how we might use it in a calculation
  - Comparing two colors:
    $H_a = 45°$, $H_b = 55°$, diff = 10°
    $H_a = 355°$, $H_b = 5°$, diff = 350° ?
    (no, they're still only 10° apart!)



Other color spaces

- HSV
  - Like HIS, but the intensity component is V for value
- CMYK
  - Cyan, Magenta, Yellow and Black
  - used in printing
- XYZ
  - “tristimulus” color signals
  - need to know the illumination properties
- Hunter L*a*b
- YCrCb
  - Color TV
  - Luminance (brightness) plus two color components



A full-color image and its various color space components (courtesy of Med Data Interactive)

a b c

FIGURE 6.39 HSI components of the RGB color image in Fig. 6.38(a). (a) Hue. (b) Saturation. (c) Intensity.

Here is a simple example of loading an image file into memory and accessing individual pixel values, using Python and OpenCV

```
import numpy as np
import cv2

# Load a color image from a Portable Network Graphics (png) file
img = cv2.imread('C:\\Data\\ichiro.png')

height = len(img)
width = len(img[0])
depth = len(img[0, 0])

# create a new image and fill with the arithmetic inverse
result  = np.zeros((height, width, depth), dtype = "uint8")
for r in range(height):
    for c in range(width):
        for p in range(depth):
            result[r, c, p] = 255-img[r, c, p];

# show images
cv2.namedWindow('INPUT', flags=cv2.WINDOW_NORMAL)
cv2.imshow('INPUT',img)
cv2.namedWindow('RESULT', flags=cv2.WINDOW_NORMAL)
cv2.imshow('RESULT',result)
cv2.waitKey(0)
cv2.destroyAllWindows()
cv2.imwrite("C:\\Data\\inverted.png", result)
```

Here is a simple example of loading an image file into memory and accessing individual pixel values, using Python and OpenCV

```
import numpy as np
import cv2

# Load a color image from a Portable Network Graphics (png) file
img = cv2.imread('C:\\Data\\ichiro.png')

height = len(img)
width = len(img[0])
depth = len(img[0, 0])

# create a new image and fill with the arithmetic inverse
result  = np.zeros((height, width, depth), dtype = "uint8")
for r in range(height):
    for c in range(width):
        for p in range(depth):
            result[r, c, p] = 255-img[r, c, p];

# show images
cv2.namedWindow('INPUT', flags=cv2.WINDOW_NORMAL)
cv2.imshow('INPUT',img)
cv2.namedWindow('RESULT', flags=cv2.WINDOW_NORMAL)
cv2.imshow('RESULT',result)
cv2.waitKey(0)
cv2.destroyAllWindows()
cv2.imwrite("C:\\Data\\inverted.png", result)
```

For all but the very simplest cases, convert raw images to floating point representation for calculation – when done, optionally convert back for display or storage

1. Load image from file, or capture from sensor
2. (optionally) convert color to grayscale, or convert to different color space
3. Convert to float32 or float64 (double)
4. Perform all calculations
5. (optionally) convert to unsigned bytes for display or storage

If processing time is absolutely crucial, we may do computation as integers – but need to be careful about negative values and overflow/underflow

IMAGE QUALITY METRICS

Bit depth and range vary by imaging application area; there is a tradeoff between precision and size/speed of processing

Grayscale (Intensity Images):

Chan.	Bits/Pix	Range	Use
1	1	[0, 1]	Binary image: documents, illustrations, fax
1	8	[0, 255]	Universal: photo, scan, print
1	12	[0, 4095]	Higher quality: photo, scan, print
1	14	[0, 16383]	Professional: photo, scan, print
1	16	[0, 65535]	Highest quality: medicine, astronomy

Color Images:

Chan.	Bits/Pix	Range	Use
3	24	$[0, 255]^3$	RGB, universal: photo, scan, print
3	36	$[0, 4096]^3$	RGB, higher quality: photo, scan, print
3	42	$[0, 16383]^3$	RGB, professional: photo, scan, print
4	32	$[0, 255]^4$	CMYK, digital prepress

Special Images:

Chan.	Bits/Pix	Range	Use
1	16	[-32768, 32767]	Integer values (pos & neg), increased range, scientific
1	32	$\pm 3.4 \times 10^{38}$	Floating-point values: medicine, astronomy
1	64	$\pm 1.8 \times 10^{308}$	Floating-point values: internal processing

All images gathered from the real world contain some level of noise (that is, variations in pixel values that have nothing to do with the scene)

- Noise also impacts the quality of an imaging system, in addition to (and on top of) resolution-related limits
- Where does noise in the image come from?
  - thermal noise in the electronics
  - imperfections in the sensor
  - lighting/emission variations
  - the quantized nature of EM (small numbers of photons)
  - gamma rays (no kidding)
- Noise is considered from a stochastic point of view
  - Modeled as a random process

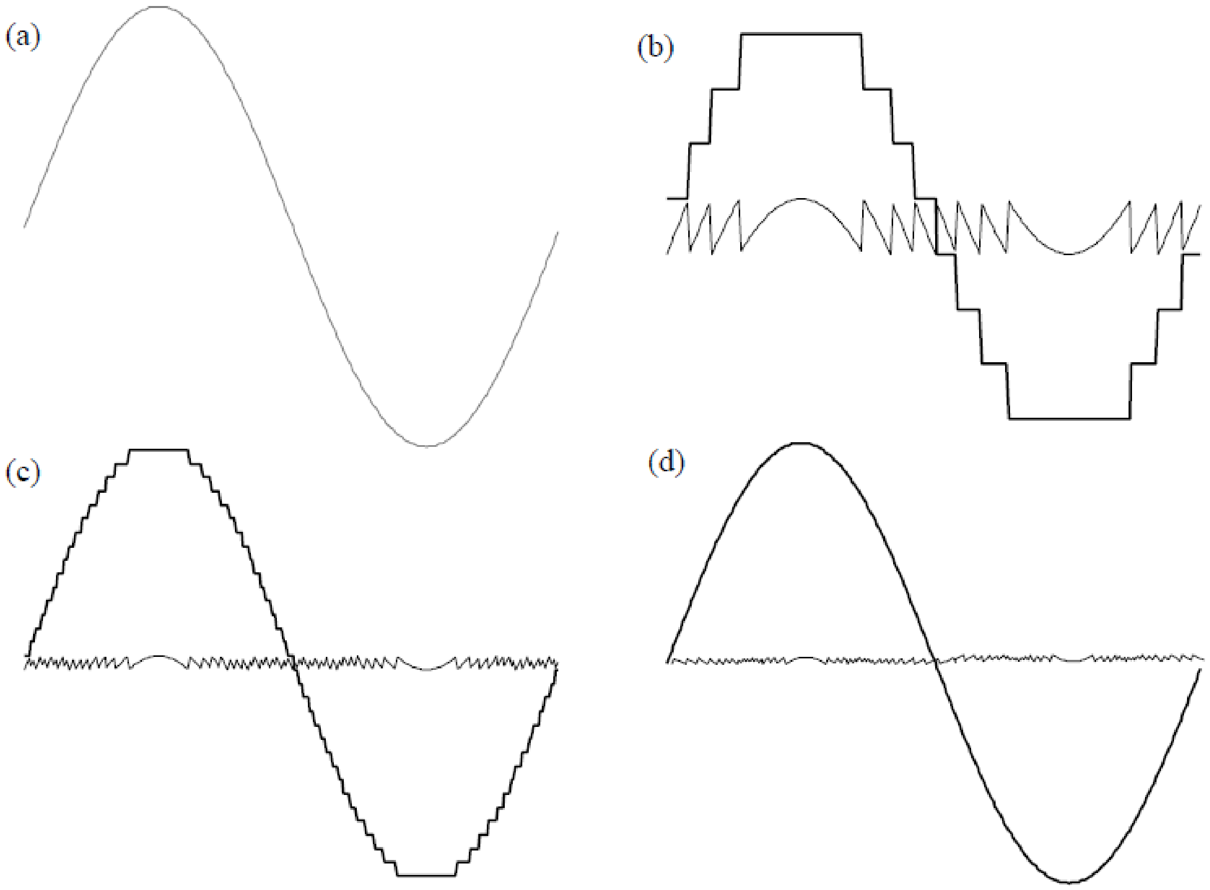
Signal to Noise Ratio (SNR) measures the overall "brightness" of the image compared to the background clutter; Contrast to Noise Ratio (CNR) measures the difference between image objects compared to clutter

- $SNR = \frac{P_{signal}}{P_{noise}} = \left(\frac{A_{signal}}{A_{noise}}\right)^2, \quad P = power, A = amplitude$

- $CNR = \frac{P_{Obj1signal} - P_{Obj2signal}}{P_{noise}} = \left(\frac{A_{Obj1signal} - A_{Obj2signal}}{A_{noise}}\right)^2$

- This image has high SNR, but low CNR

Analog to Digital Conversion introduces loss of fidelity due to reduction to a fixed number of levels – this is *quantization noise*



An original signal (a) digitized by 3-bit (b), 5-bit (c) and 6-bit (d) Analog to Digital convertors.

The lower amplitude signal is the quantization noise,

$$v_{qnoise} = v_{act} - v_{conv}$$

Dynamic Range of a system is the number of valid signal levels represented in the final image; conversion resolution relates this to signal input

- Consider an Analog to Digital convertor with 14 output bits
- The dynamic range is $2^{14} = 16{,}384$ levels
- The conversion resolution for a 5-volt full-scale input is:
  $5V \frac{1}{16{,}384\ levels} = 0.3\ millivolts$
- (We assume that the amplifiers and transmission has a low enough noise level that these levels are useful data)
  - What S/N ratio is required?

$$SNR = \frac{\mu_{sig}}{\sigma_{sig}} = \frac{5v}{0.3mV} = 16{,}384\ \ldots\ \text{or so}$$

Today’s Objectives

- An Image in Memory
- Color Image Sensing
  - Bayer Filters and Demosaicing
- A Color Image in Memory
- Common Color spaces
  - Red – Green – Blue (RGB)
  - Hue – Saturation – Intensity (HSI)
  - RGB ←→ HSI Conversion
- Other Color Spaces
- Image Quality Metrics