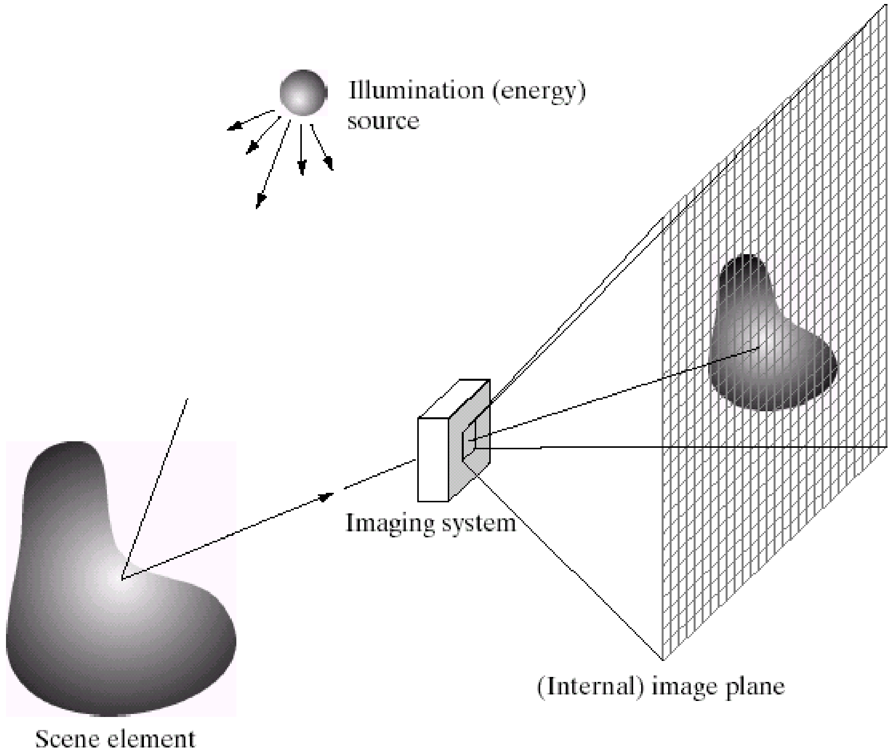
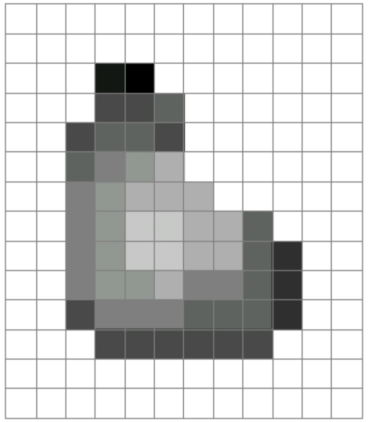
Image formation, light, color

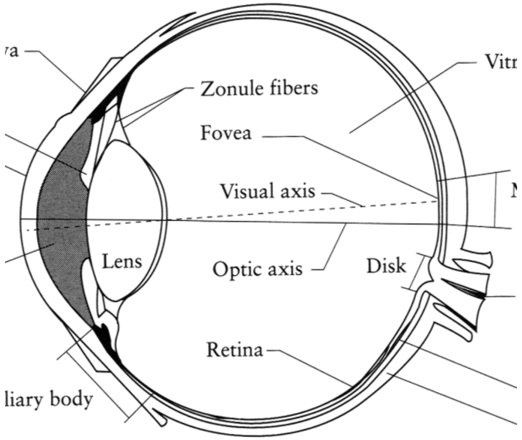
Image Formation



Film

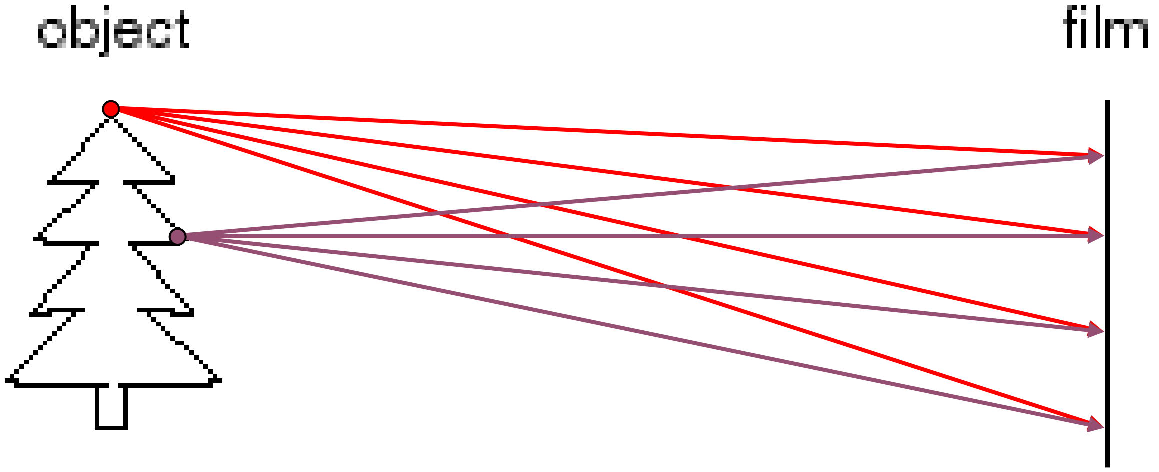


Digital Camera



The Eye

Image formation

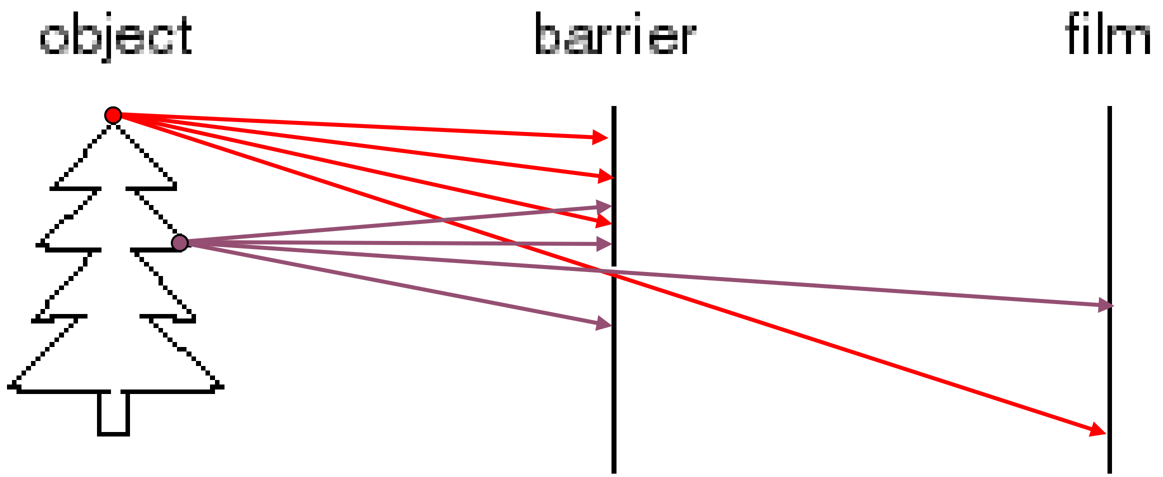


Let’s design a camera

- Idea 1: put a piece of film in front of an object
- Do we get a reasonable image?

Slide source: Seitz

Pinhole camera

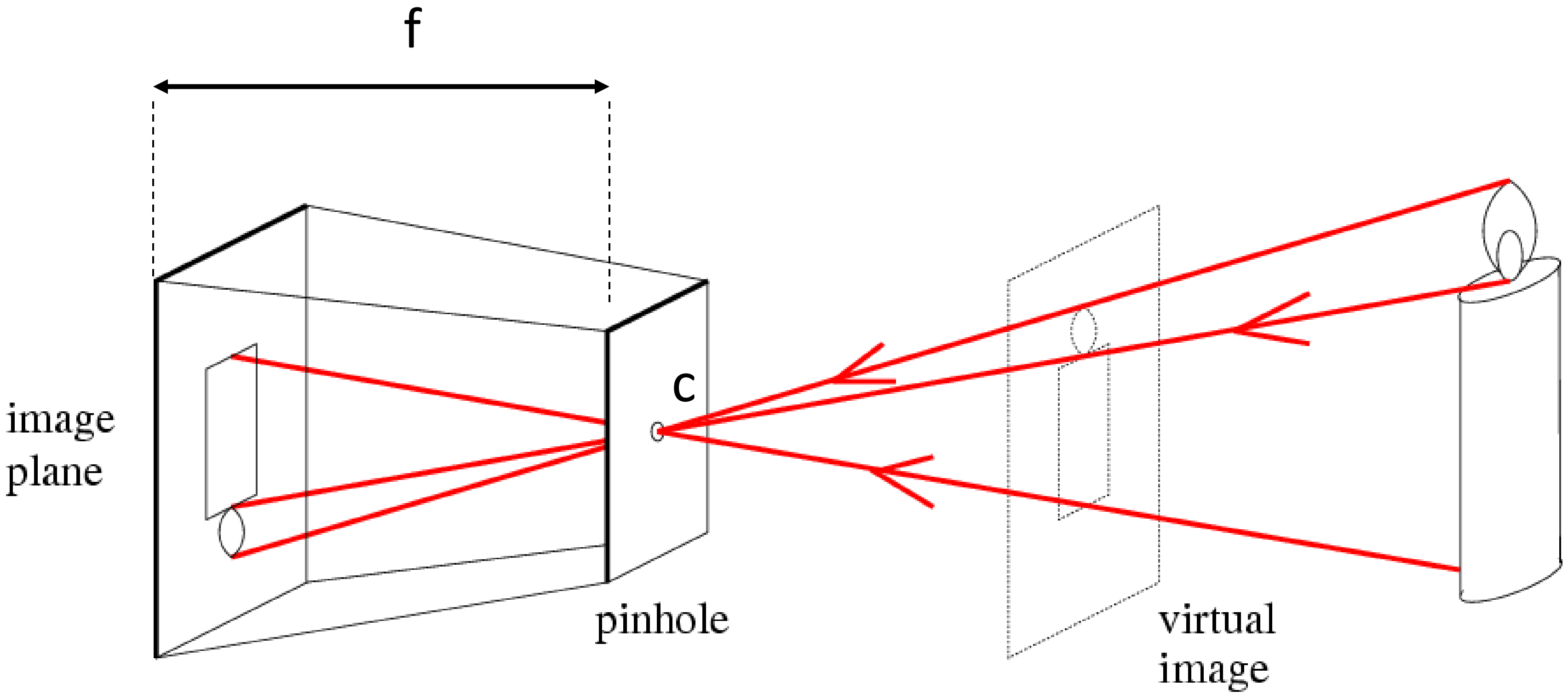


Idea 2: add a barrier to block off most of the rays

- This reduces blurring
- The opening known as the **aperture**

Slide source: Seitz

Pinhole camera

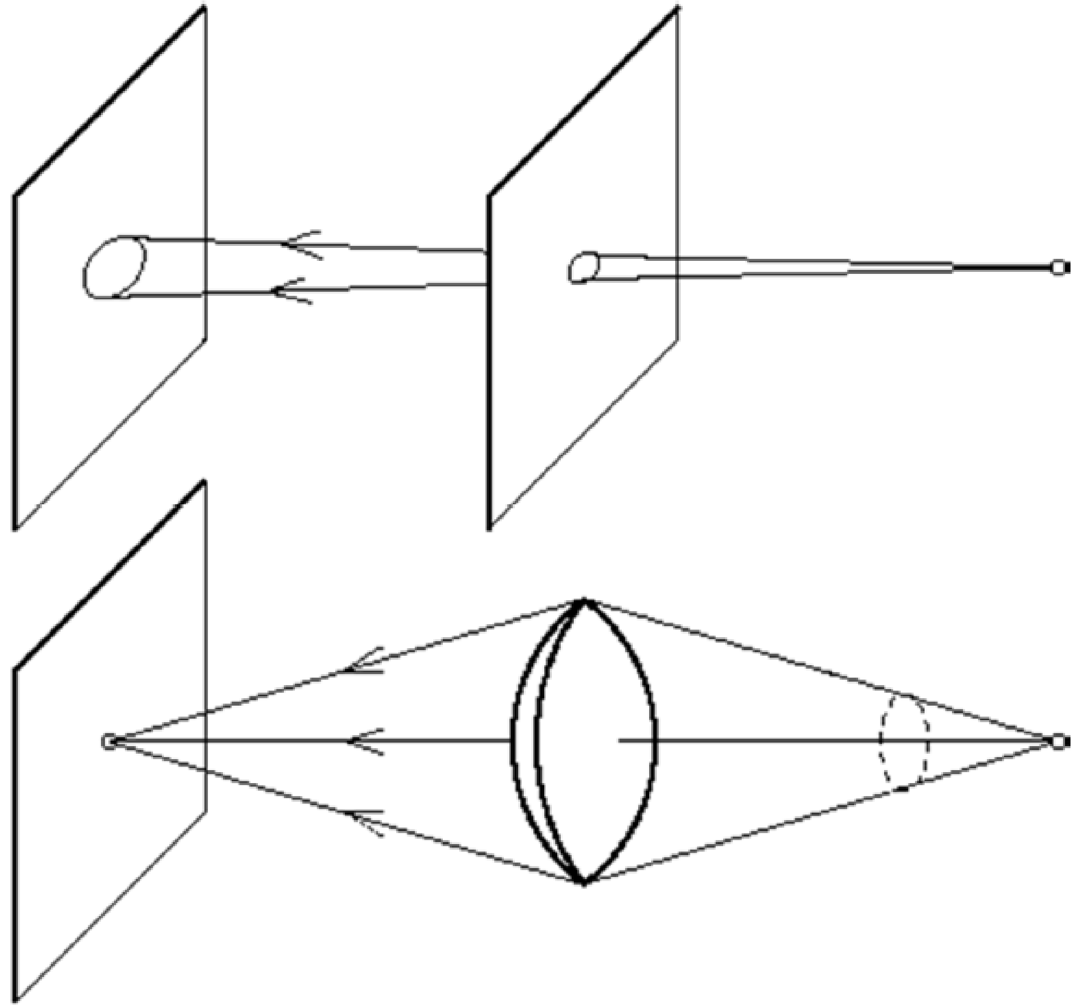


f = focal length

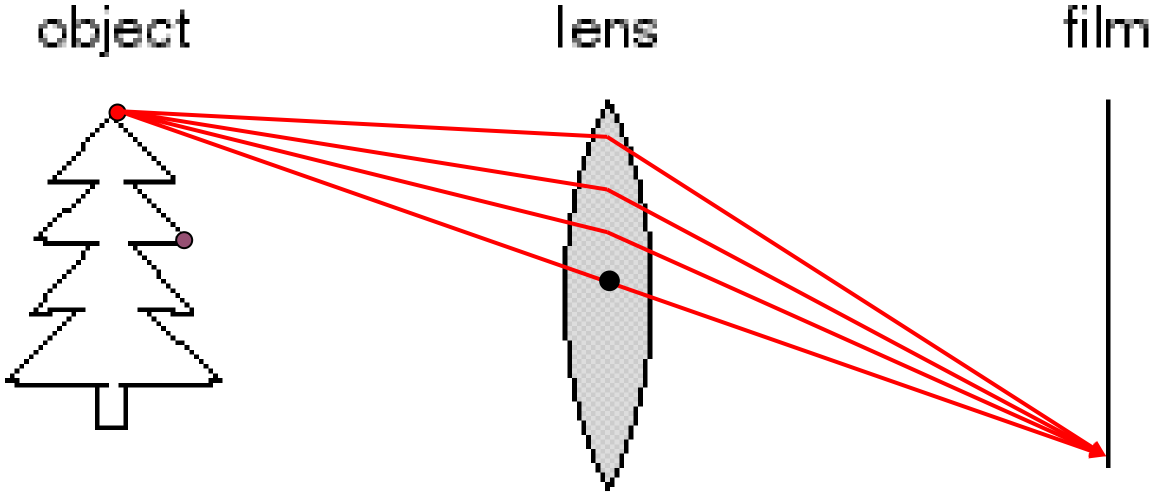
c = center of the camera

Figure from Forsyth

The reason for lenses



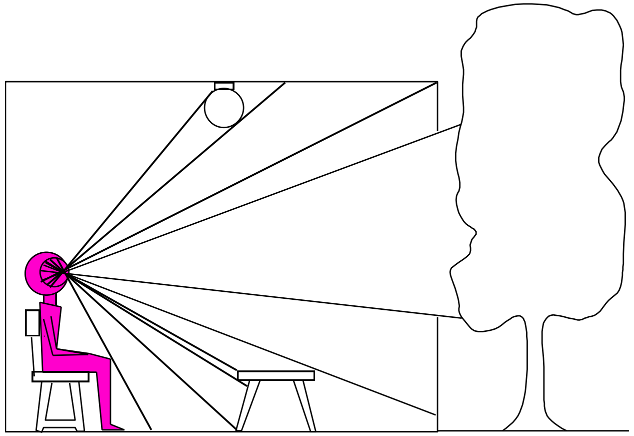
Adding a lens



- A lens focuses light onto the film

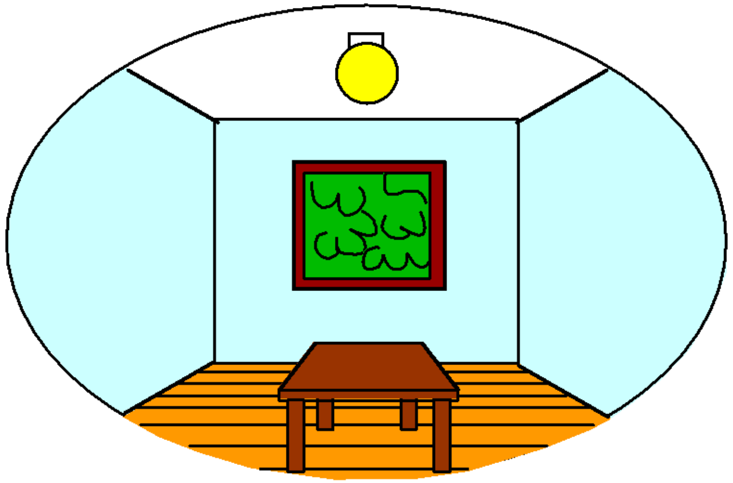
Dimensionality Reduction Machine (3D to 2D)

3D world



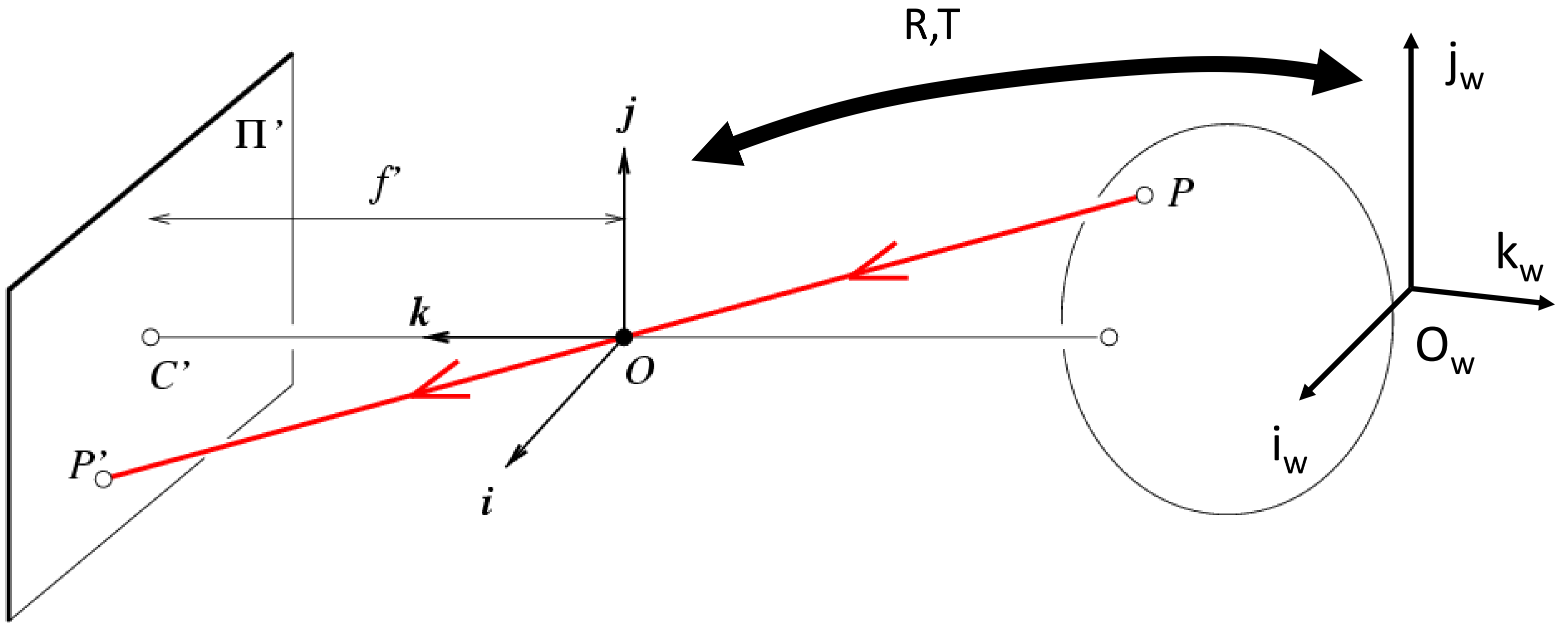
Point of observation

2D image



Projection matrix

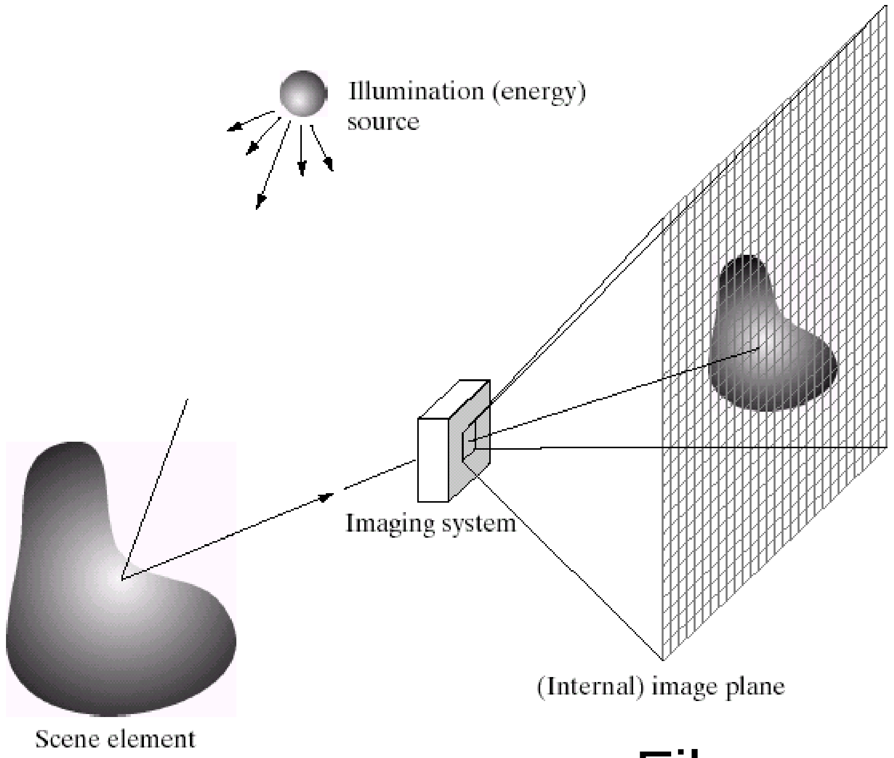
Slide Credit: Saverese



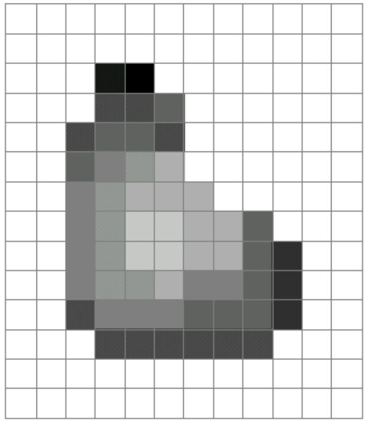
$$\mathbf{x} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{X}$$

x: Image Coordinates: (u,v,1)
K: Intrinsic Matrix (3x3)
R: Rotation (3x3)
t: Translation (3x1)
X: World Coordinates: (X,Y,Z,1)

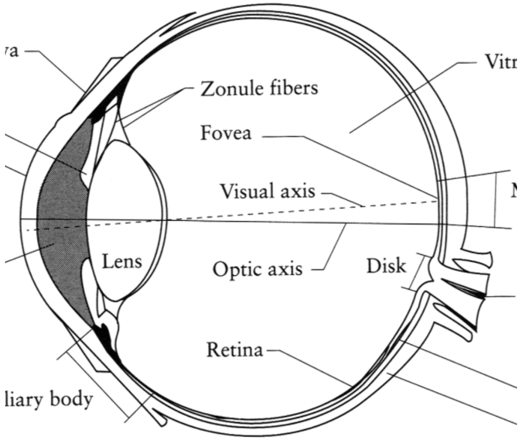
Image Formation



Film



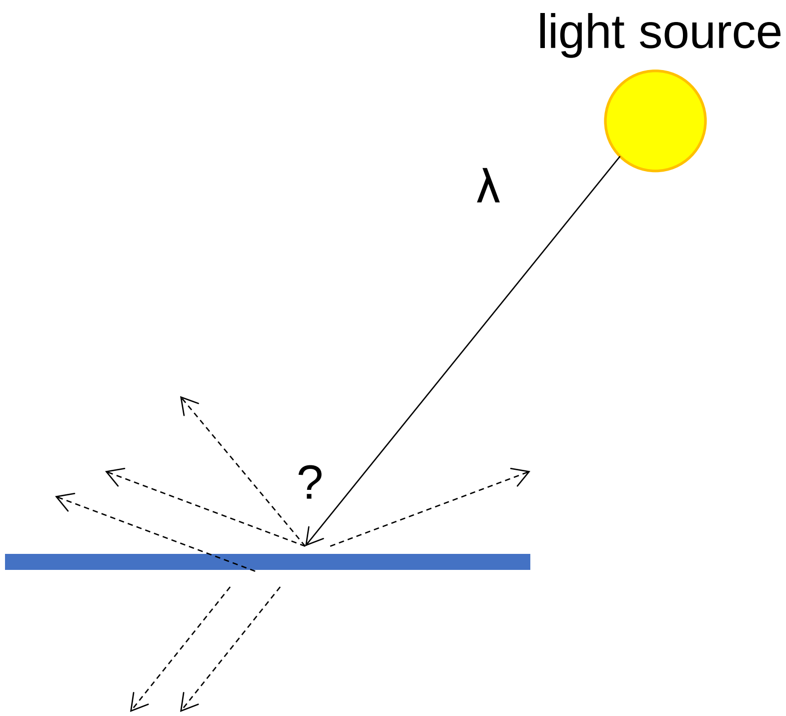
Digital Camera



The Eye

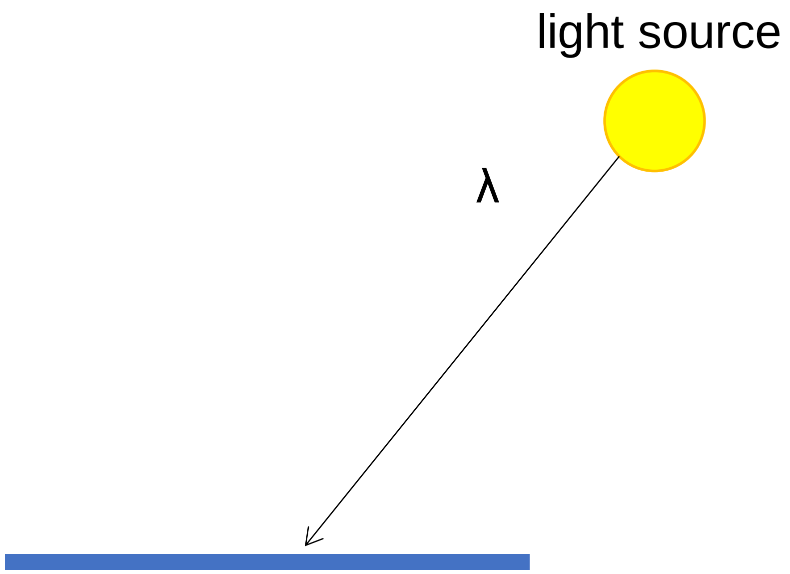
A photon’s life choices

- Absorption
- Diffusion
- Reflection
- Transparency
- Refraction



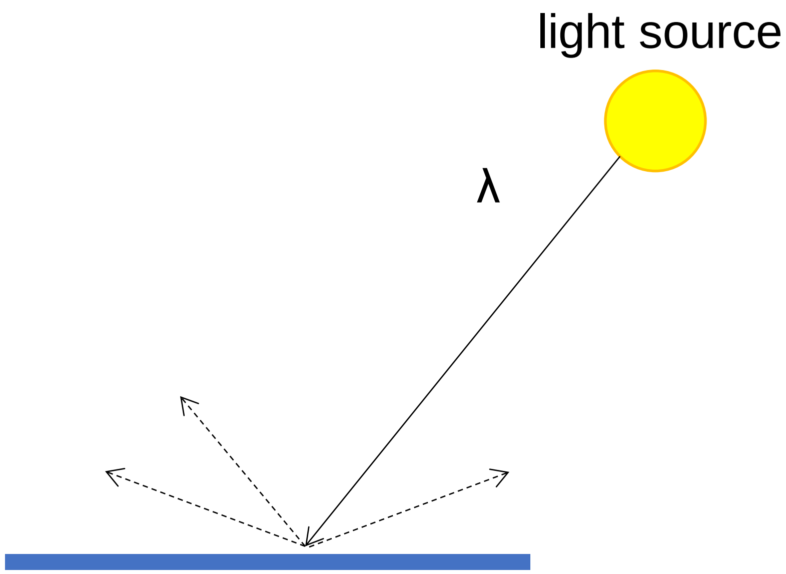
A photon's life choices

- **Absorption**
- Diffusion
- Reflection
- Transparency
- Refraction



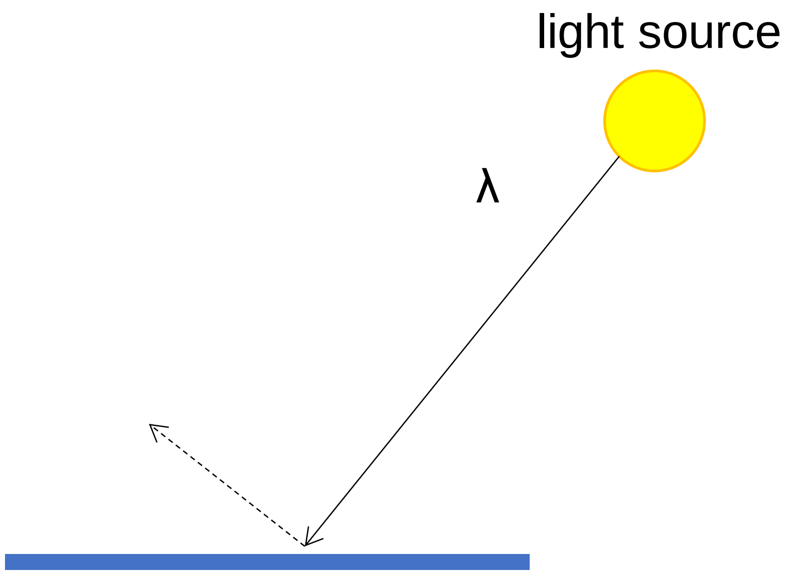
A photon’s life choices

- Absorption
- **Diffuse Reflection**
- Reflection
- Transparency
- Refraction



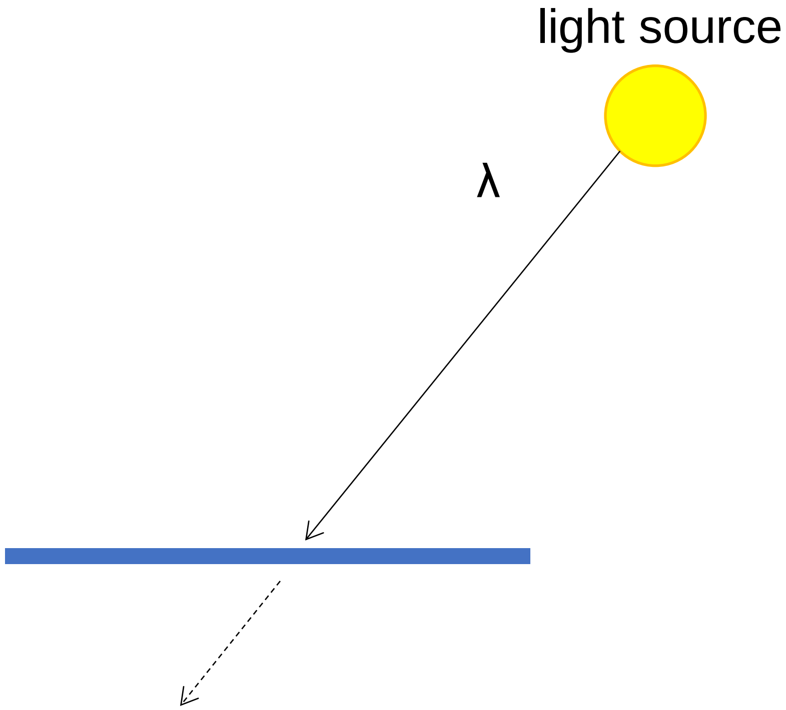
A photon's life choices

- Absorption
- Diffusion
- **Specular Reflection**
- Transparency
- Refraction



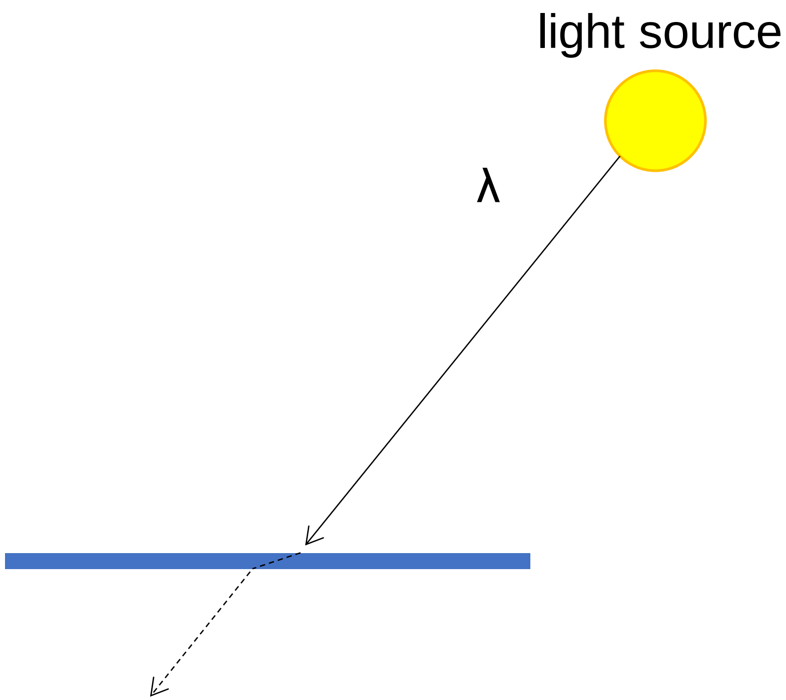
A photon’s life choices

- Absorption
- Diffusion
- Reflection
- **Transparency**
- Refraction



A photon’s life choices

- Absorption
- Diffusion
- Reflection
- Transparency
- **Refraction**



Lambertian Reflectance

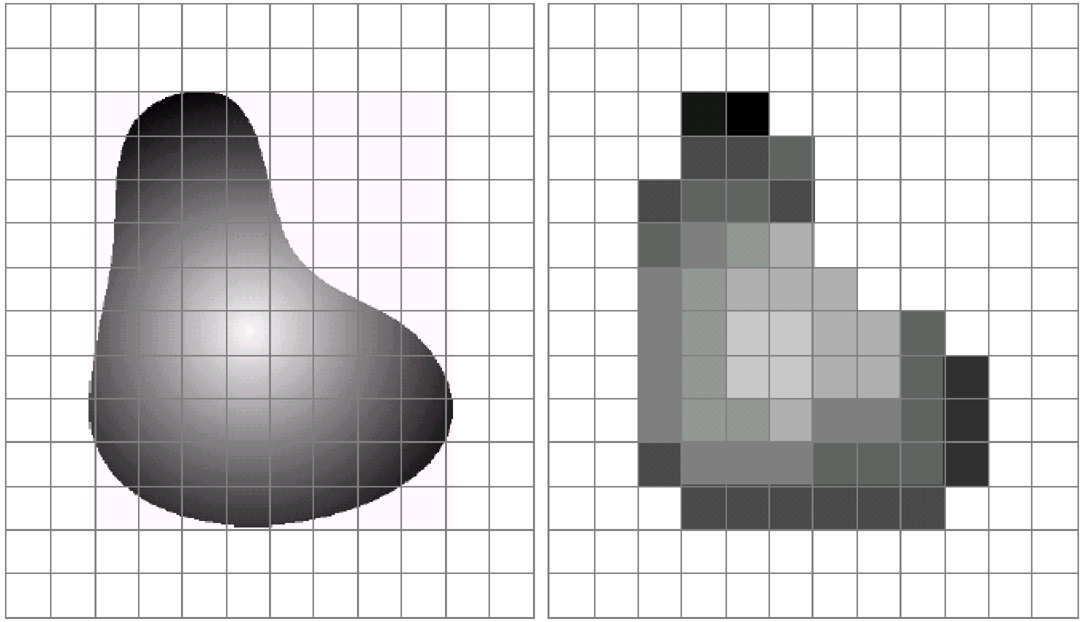
- In computer vision, surfaces are often assumed to be ideal diffuse reflectors.

Digital camera

- A digital camera replaces film with a sensor array
  - Each cell in the array is light-sensitive diode that converts photons to electrons
  - http://electronics.howstuffworks.com/digital-camera.htm

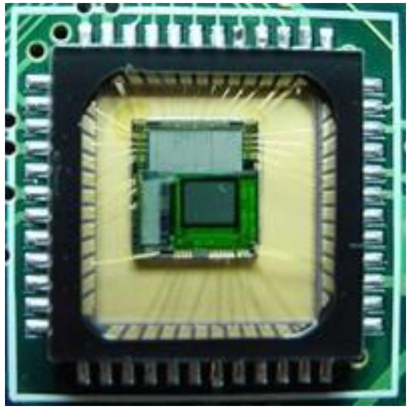
Slide by Steve Seitz

Sensor Array



a b

FIGURE 2.17 (a) Continuos image projected onto a sensor array. (b) Result of image sampling and quantization.



CMOS sensor

Sampling and Quantization

- In order to become suitable for computer processing, an image function f(x, y) must be digitized both spatially and in amplitude. This can be done using two steps:
  - **Sampling**: Image digitization means that the function f(x, y) is sampled into a matrix with M rows and N columns.
  - **Quantization**: The image quantization assigns to each sample an integer value. The continuous range of the image function f(x, y) is split into k intervals.

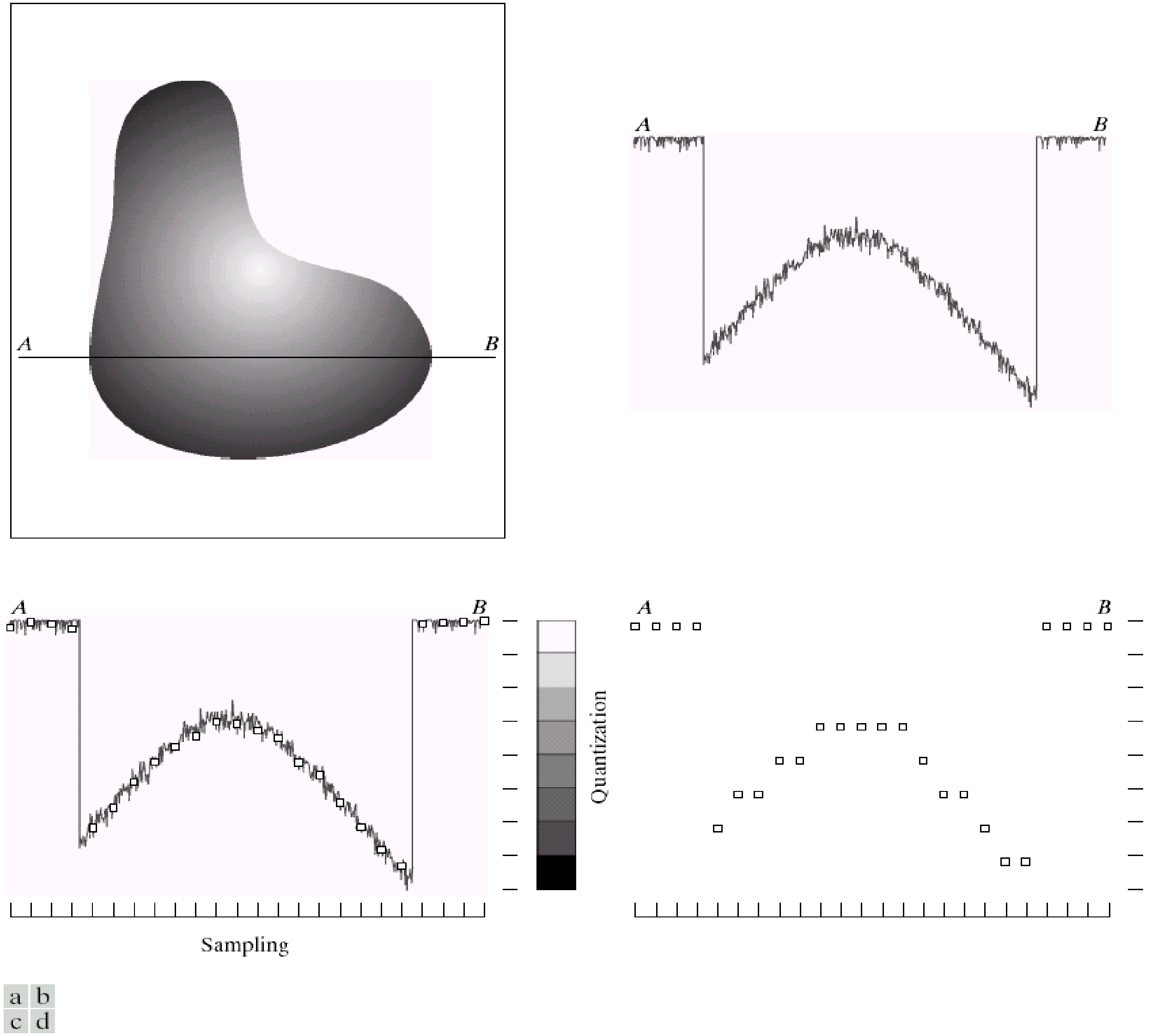
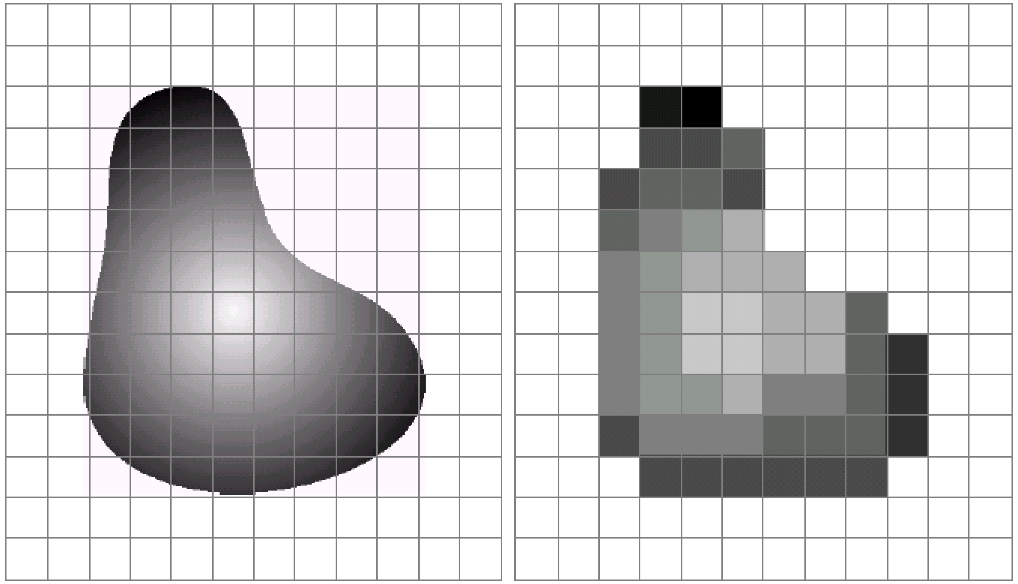


FIGURE 2.16 Generating a digital image. (a) Continuous image. (b) A scan line from A to B in the continuous image, used to illustrate the concepts of sampling and quantization. (c) Sampling and quantization. (d) Digital scan line.

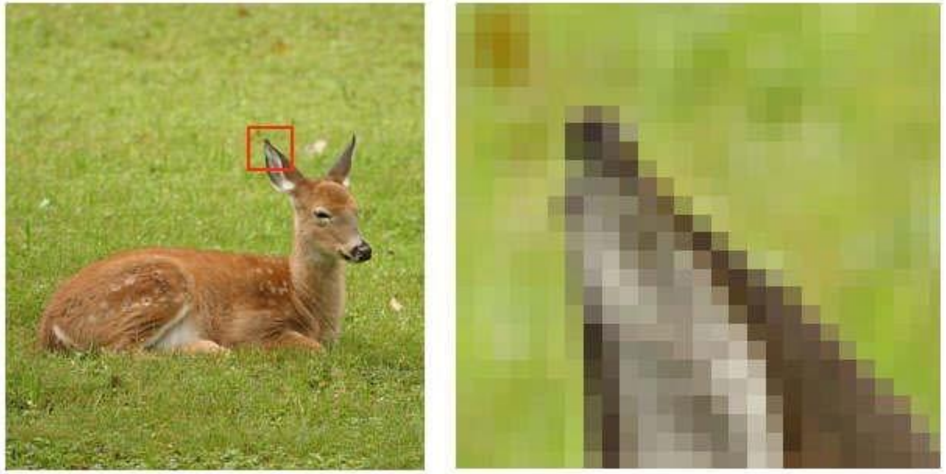
…

- Most digital image processing devices use quantization into k equal intervals.
- If *b* bits are used ... the number of ***brightness*** levels is $k=2^b$
    - 1 bit: k=2 colors; for example: black and white
    - 2 bits: k= 4 colors.
    - 8 bits: k = 256 colors; for example: grey level image (0: black, 255: white)
- The number of quantization levels should be high enough for human perception of fine shading details in the image.



a b

FIGURE 2.17 (a) Continuos image projected onto a sensor array. (b) Result of image sampling and quantization.



a b c
d e f

FIGURE 2.20 (a) 1024 × 1024, 8-bit image. (b) 512 × 512 image resampled into 1024 × 1024 pixels by row and column duplication. (c) through (f) 256 × 256, 128 × 128, 64 × 64, and 32 × 32 images resampled into 1024 × 1024 pixels.

The Resolution of an Image Source

The resolution of an image can be specified in terms of 2 quantities:

- **Spatial resolution** The column (C) by row (R) dimensions of the image define the number of pixels used to cover the visual space captured by the image.
- **Bit resolution** This defines the number of possible intensity/color values that a pixel may .
  - The bit resolution is commonly quoted as the number of binary bits required for storage at a given quantization level, e.g., binary is 2 bit, grey-scale is 8 bit and color (most commonly) is 24 bit.

8-bit Greyscale Images (Intensity Images)

W=100

H=100

row

156	159	158	155 …
160	154	157	158 …
156	159	158	155 …
160	154	157	158 …
…	…	…	…

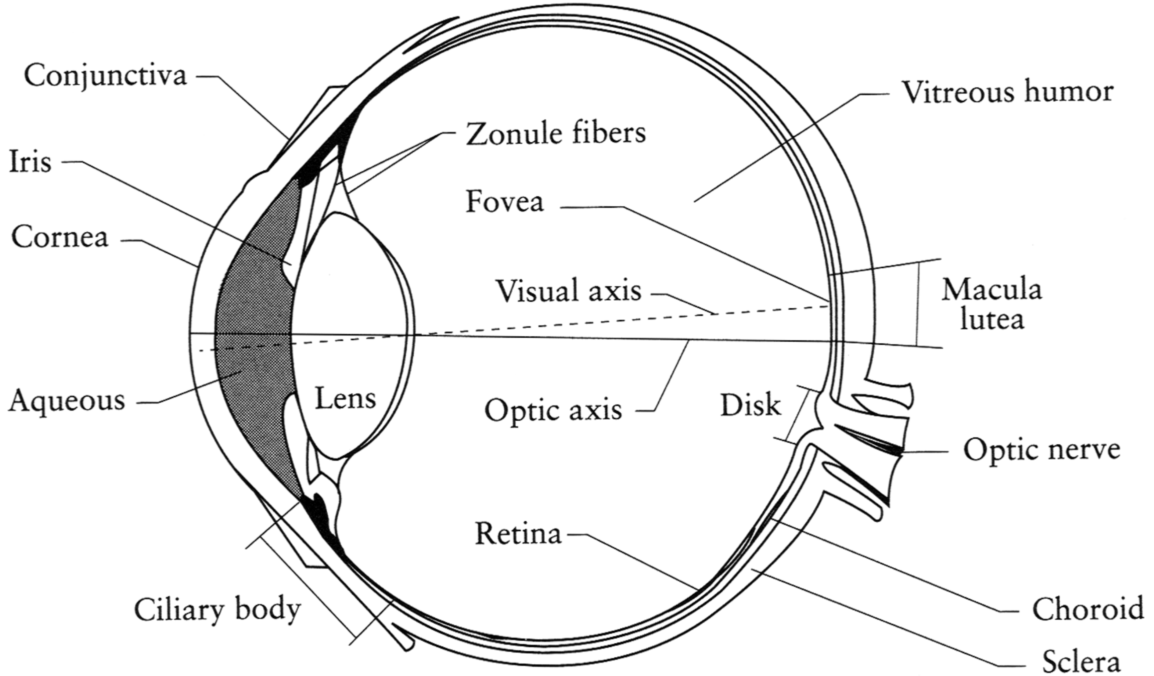
column

Gray-level: 8 bits or 1 byte per pixel

<u>Raw image data</u>
No header information, just pack the pixel values in a rastering scanning order

The size of a raw image file is HW bytes (H,W are the height and width of the image)

The Eye



- The human eye is a camera!
  - **Iris** - colored annulus with radial muscles
  - **Pupil** - the hole (aperture) whose size is controlled by the iris
  - What's the "film"?
    - photoreceptor cells (rods and cones) in the **retina**

Slide by Steve Seitz

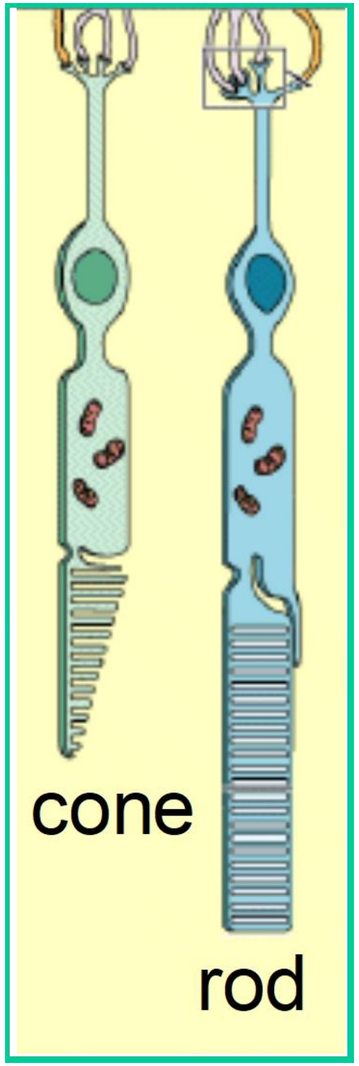
Two types of light-sensitive receptors

Cones

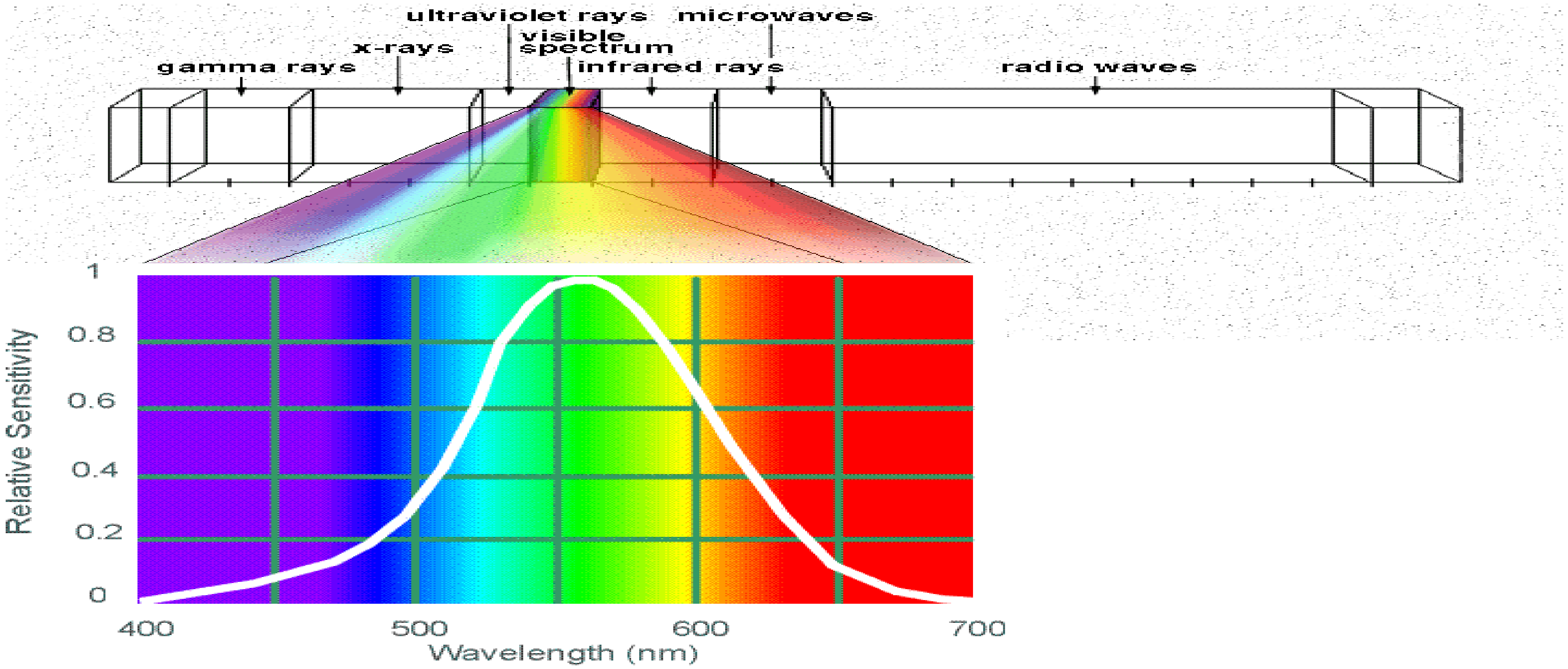
cone-shaped
less sensitive
operate in high light
color vision

Rods

rod-shaped
highly sensitive
operate at night
gray-scale vision



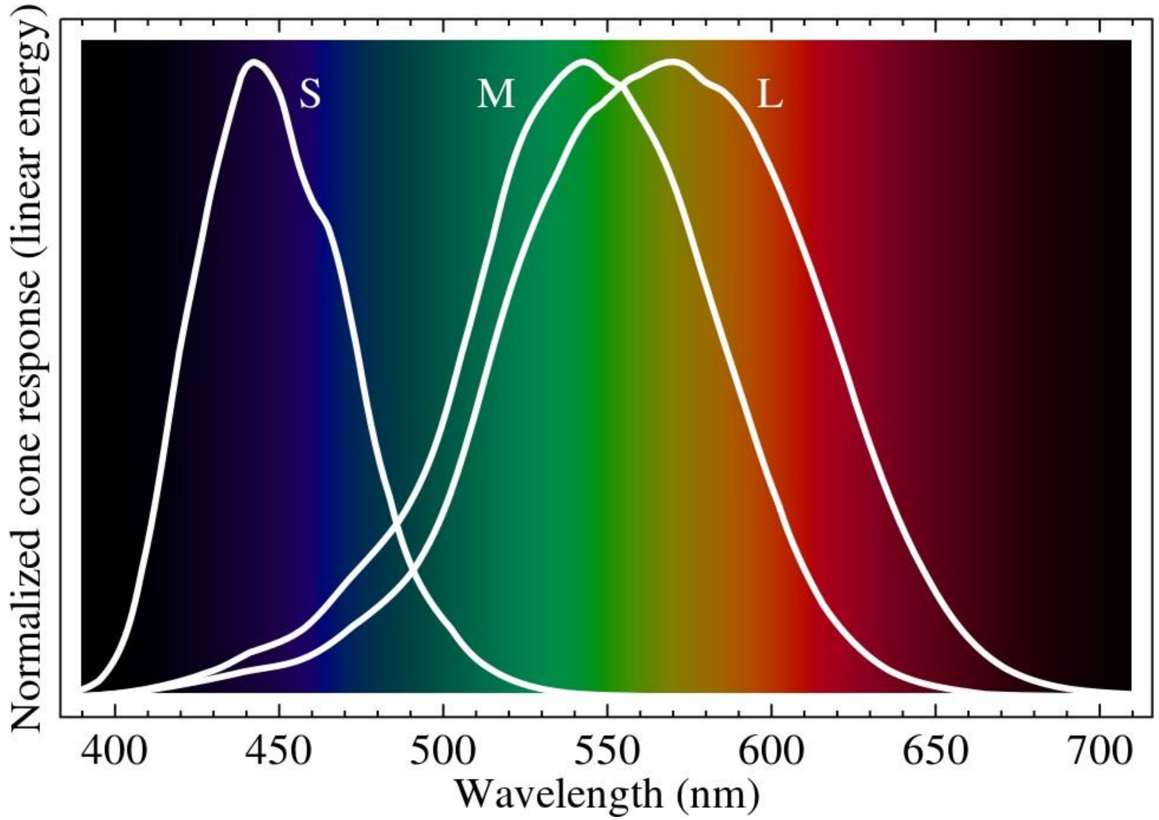
Electromagnetic Spectrum



Human Luminance Sensitivity Function

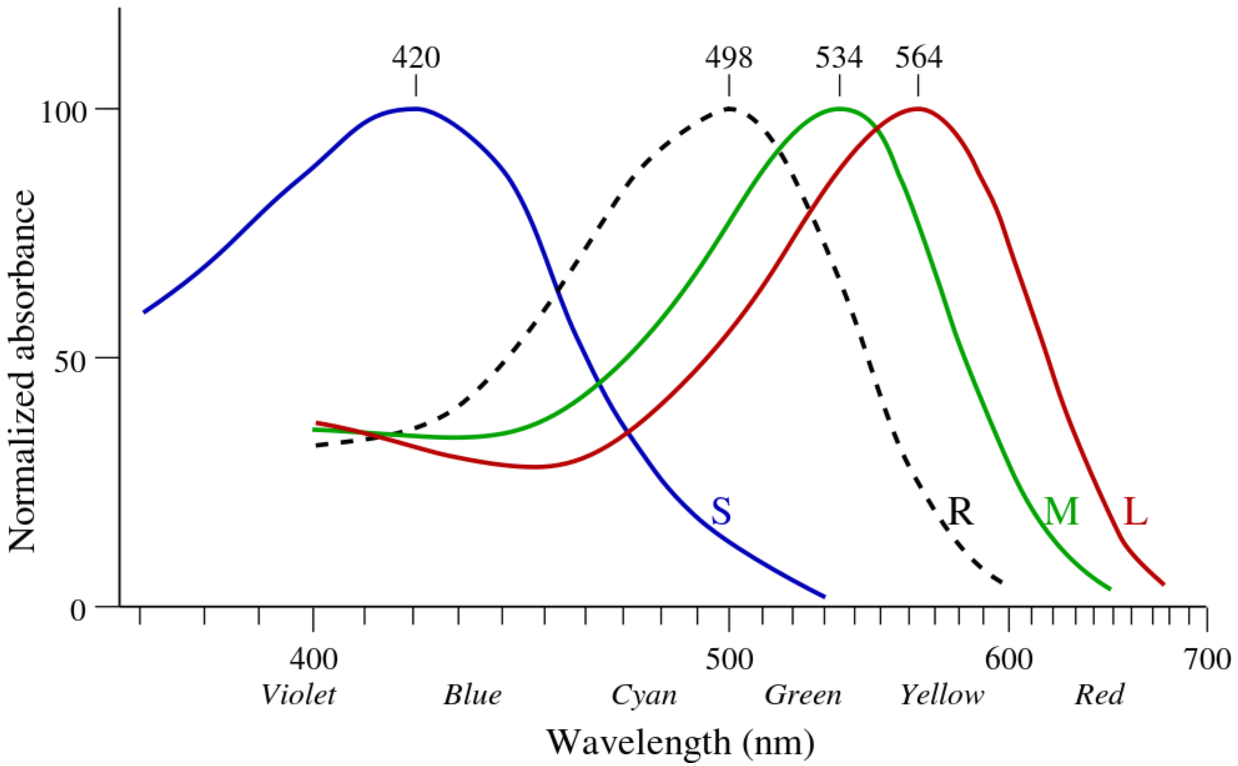
http://www.yorku.ca/eye/photopik.htm

- Humans have **3 types of cone sensors** (cones):
    - Cones that absorb long-wavelength light (L)
    - Cones that absorb middle-wavelength light (M)
    - Cones that absorb short-wavelength light (S)
- Colors are perceived by the interaction of at least 2 types of cones

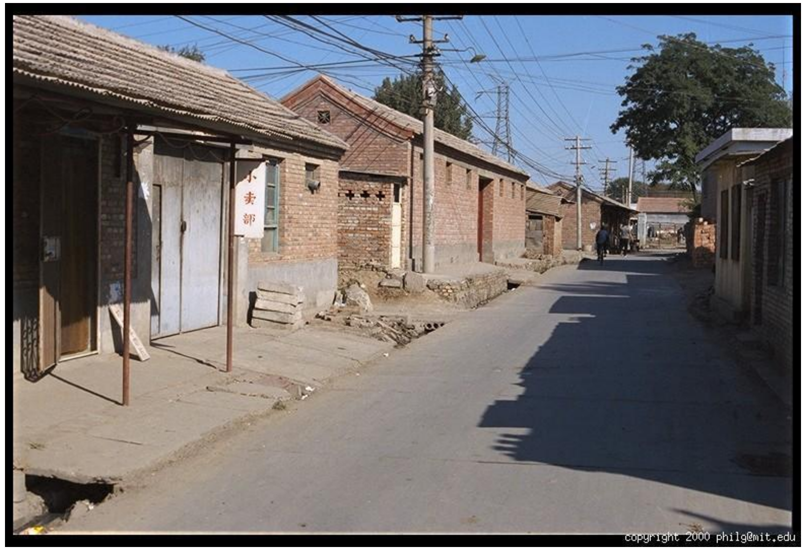


ROD CELLS

- Are in the retina of the eye
- Are **sensitive to less intense light** than the others
- Concentrated on the **outer** edges
- Used in **peripheral vision** and in **low light**



Color Image



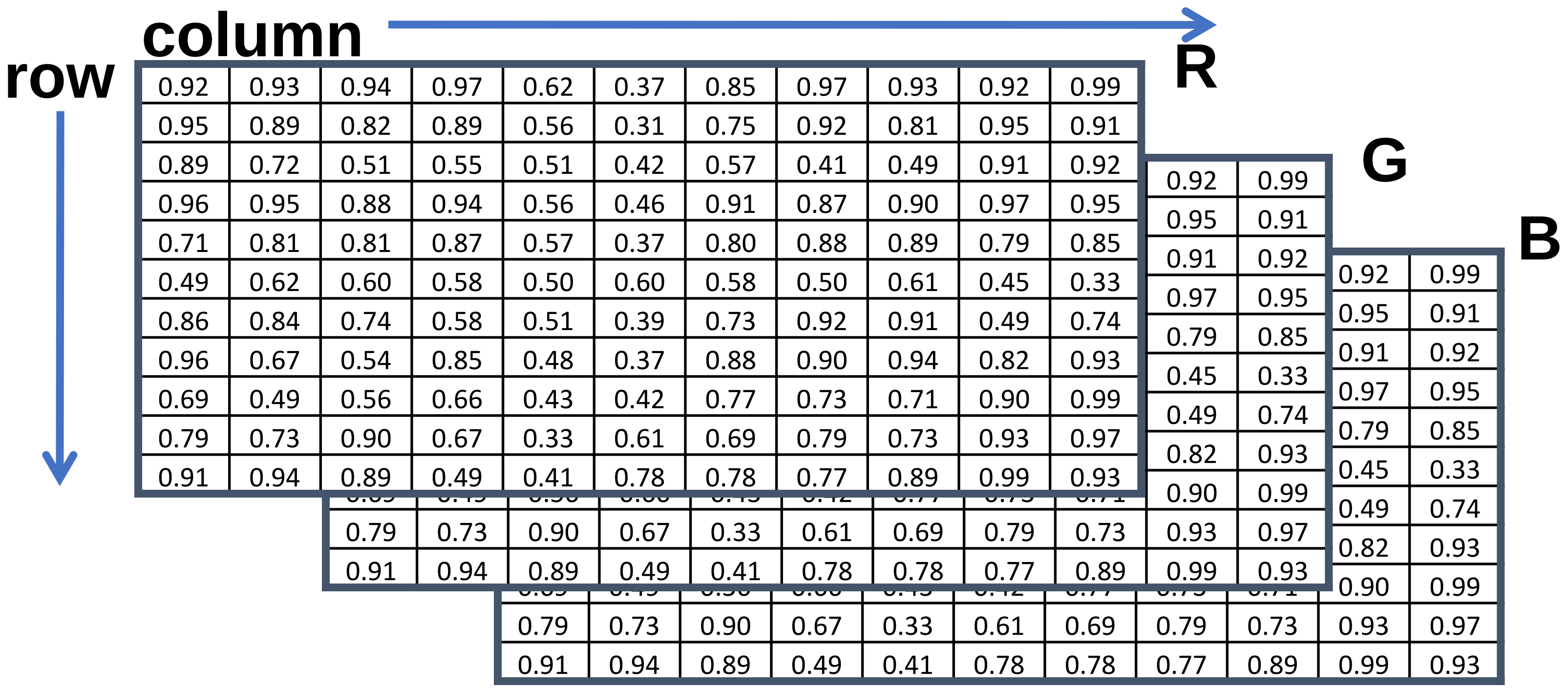
R
(G=0,B=0)

G
(R=0,B=0)

B
(R=0,G=0)

Images in Matlab

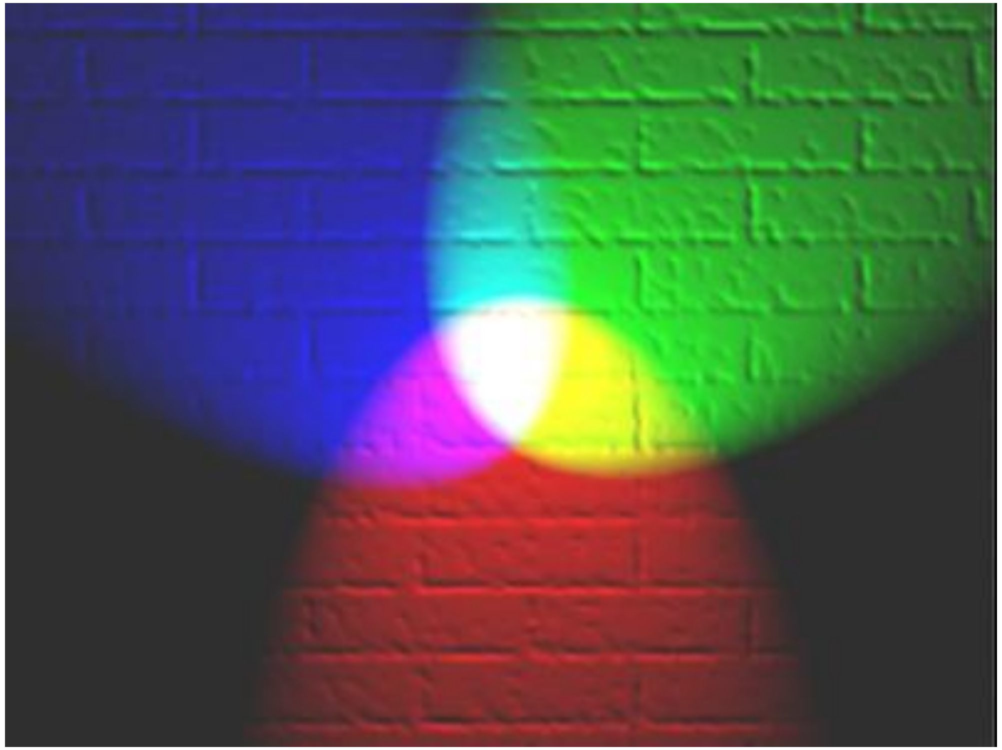
- Images represented as a matrix
- Suppose we have a NxM RGB image called "im"
  - im(1,1,1) = top-left pixel value in R-channel
  - im(y, x, b) = y pixels down, x pixels to right in the b^{th} channel
  - im(N, M, 3) = bottom-right pixel in B-channel
- imread(filename) returns a uint8 image (values 0 to 255)
  - Convert to double format (values 0 to 1) with im2double



0.92	0.93	0.94	0.97	0.62	0.37	0.85	0.97	0.93	0.92	0.99
0.95	0.89	0.82	0.89	0.56	0.31	0.75	0.92	0.81	0.95	0.91
0.89	0.72	0.51	0.55	0.51	0.42	0.57	0.41	0.49	0.91	0.92
0.96	0.95	0.88	0.94	0.56	0.46	0.91	0.87	0.90	0.97	0.95
0.71	0.81	0.81	0.87	0.57	0.37	0.80	0.88	0.89	0.79	0.85
0.49	0.62	0.60	0.58	0.50	0.60	0.58	0.50	0.61	0.45	0.33
0.86	0.84	0.74	0.58	0.51	0.39	0.73	0.92	0.91	0.49	0.74
0.96	0.67	0.54	0.85	0.48	0.37	0.88	0.90	0.94	0.82	0.93
0.69	0.49	0.56	0.66	0.43	0.42	0.77	0.73	0.71	0.90	0.99
0.79	0.73	0.90	0.67	0.33	0.61	0.69	0.79	0.73	0.93	0.97
0.91	0.94	0.89	0.49	0.41	0.78	0.78	0.77	0.89	0.99	0.93

Color spaces

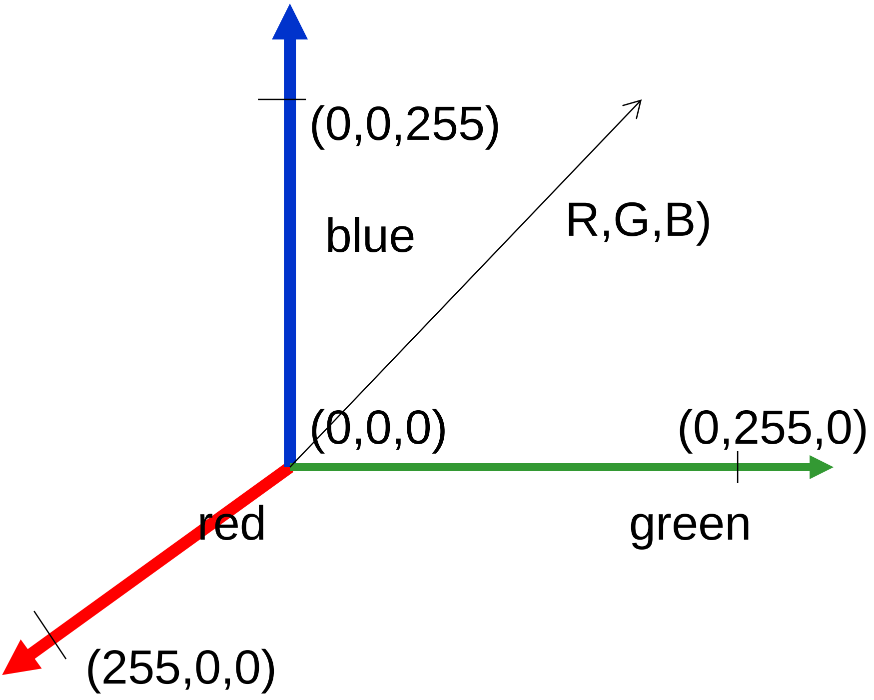
- How can we represent color?

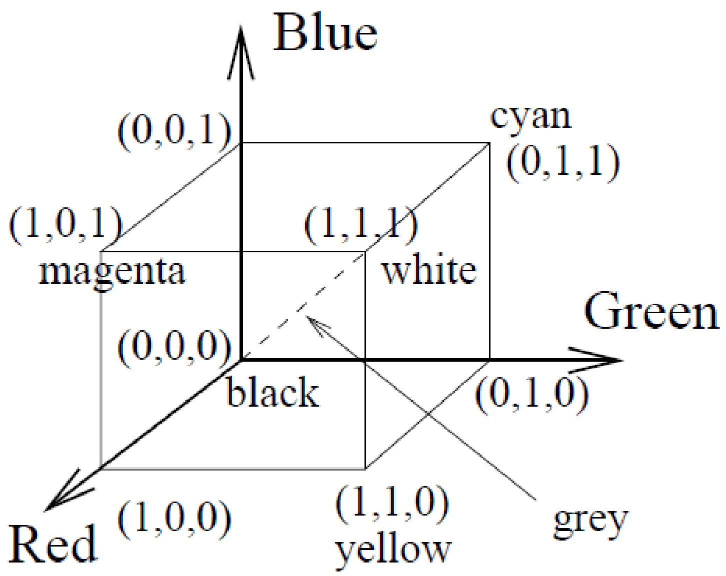


http://en.wikipedia.org/wiki/File:RGB_illumination.jpg

RGB Color Space

Absolute





Normalized

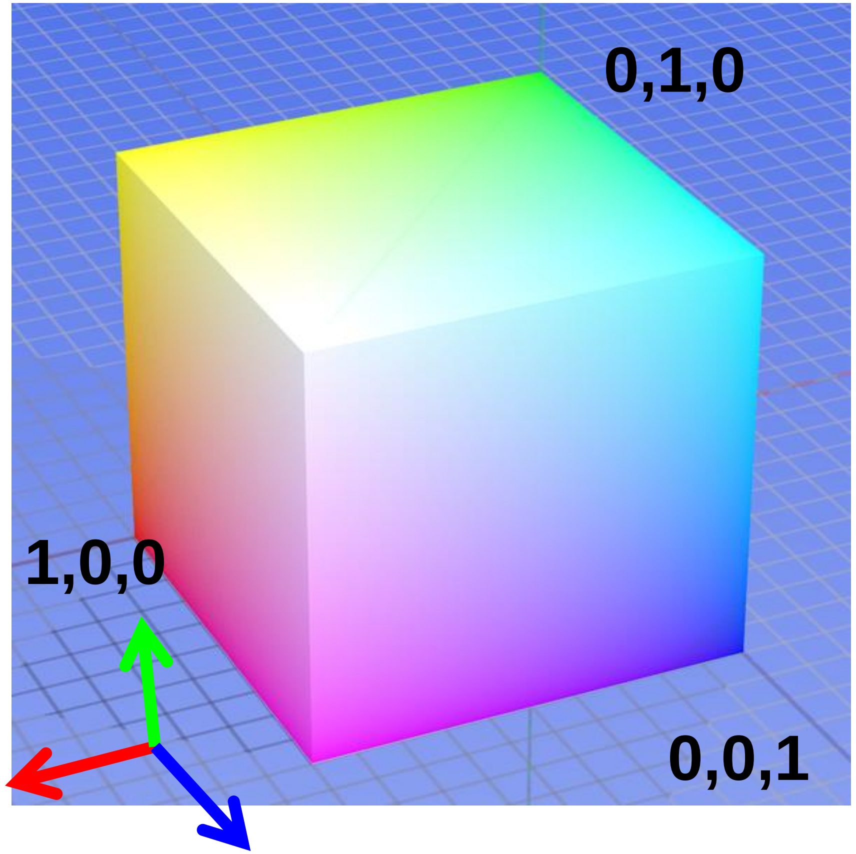
Normalized red **r = R/(R+G+B)**

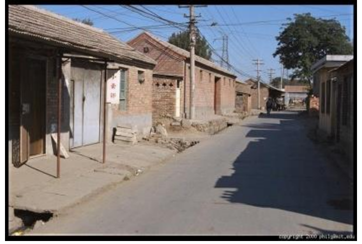
Normalized green **g = G/(R+G+B)**

Normalized blue **b = B/(R+G+B)**

Color spaces: RGB

Default color space





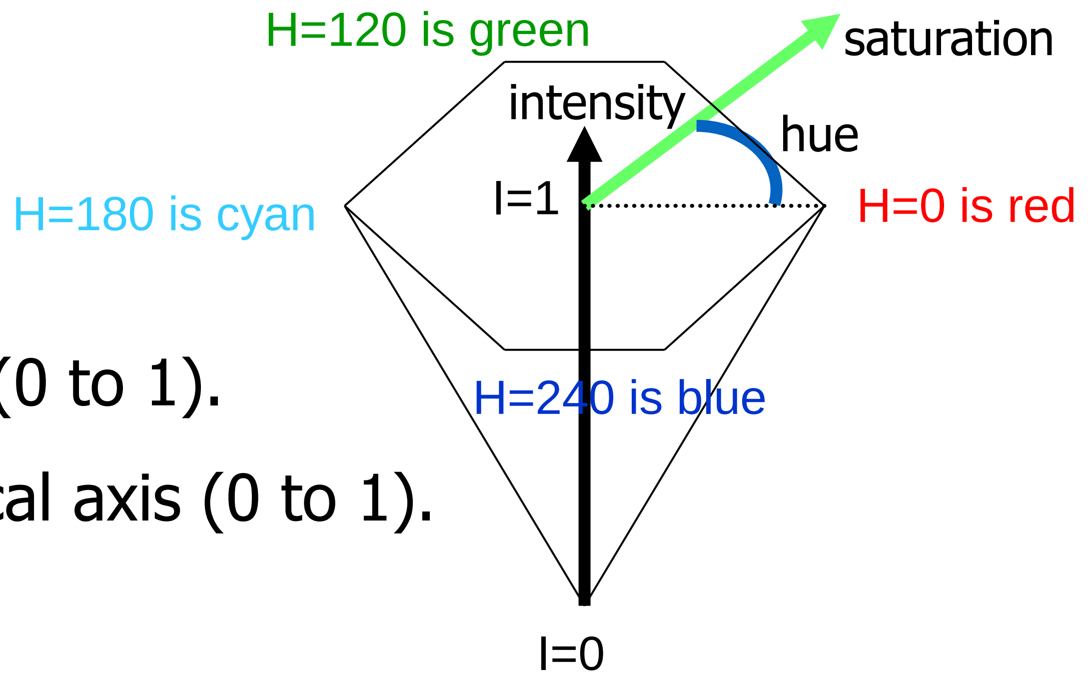
R
(G=0,B=0)

G
(R=0,B=0)

B
(R=0,G=0)

Image from: http://en.wikipedia.org/wiki/File:RGB_color_solid_cube.png

Color hexagon for HSI (HSV)



- **Hue** is encoded as an angle (0 to 2π).
- **Saturation** is the distance to the vertical axis (0 to 1).
- **Intensity/Value** is the height along the vertical axis (0 to 1).

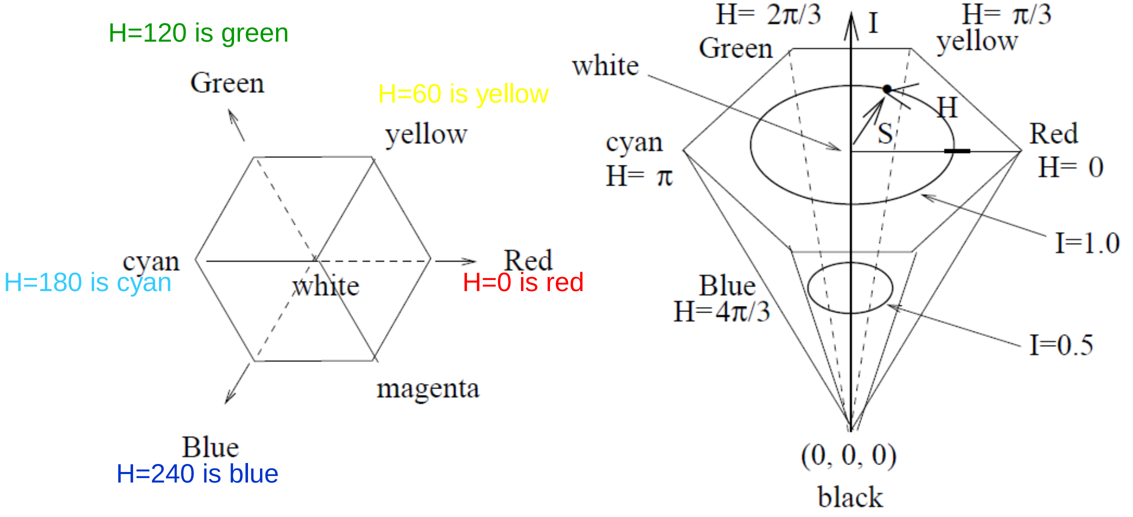
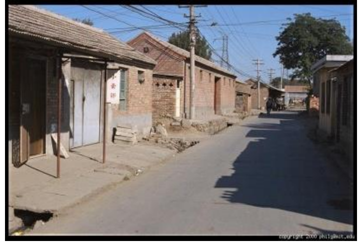
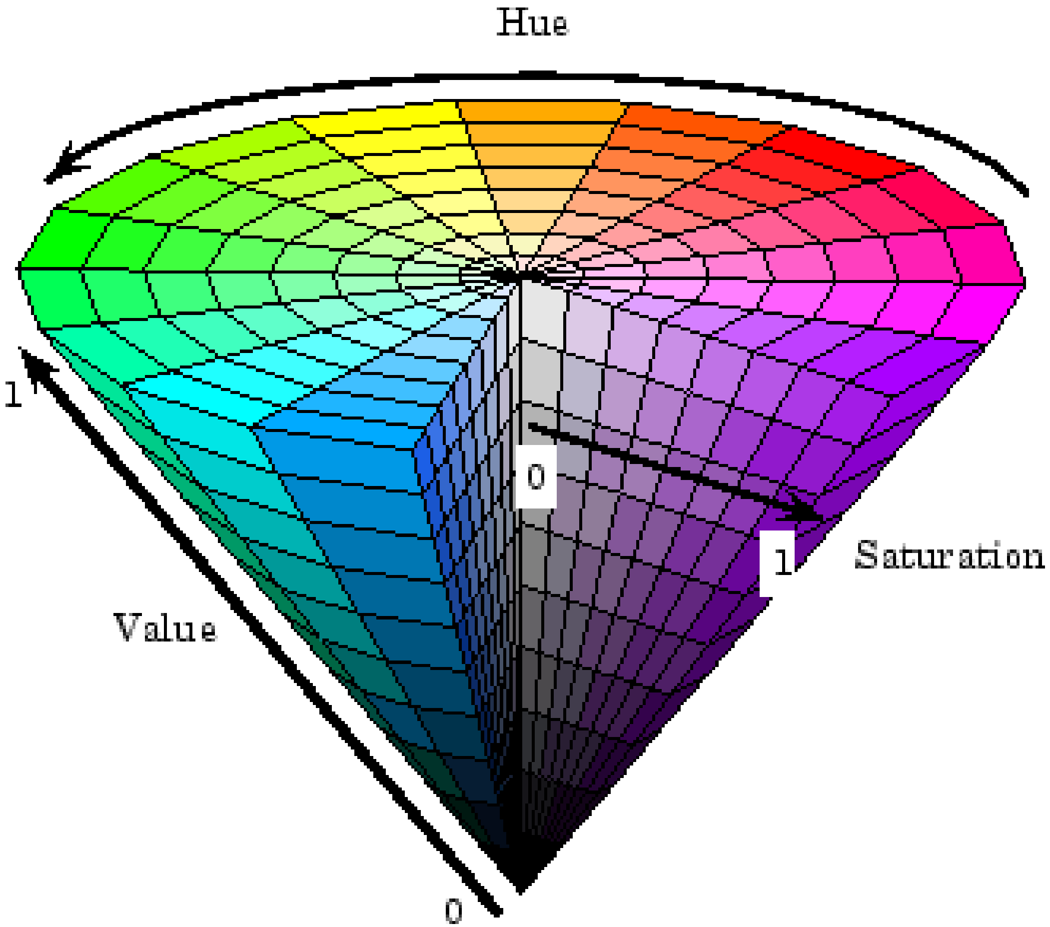


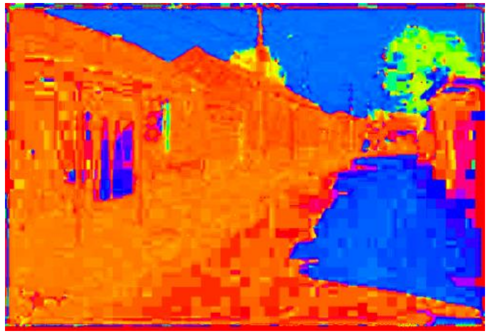
Figure 6.8: Color hexacone for HSI representation. At the left is a projection of the RGB cube perpendicular to the diagonal from (0, 0, 0) to (1, 1, 1): color names now appear at the vertices of a hexagon. At the right is a hexacone representing colors in HSI coordinates: intensity (I) is the vertical axis; hue (H) is an angle from 0 to 2π with RED at 0.0; saturation (S) ranges from 0 to 1 according to how pure, or unlike white, the color is with S=0.0 corresponding to the I-axis.

Color spaces: HSV (or HSI)



Intuitive color space





H
(S=1,V=1)

S
(H=1,V=1)

V
(H=1,S=0)

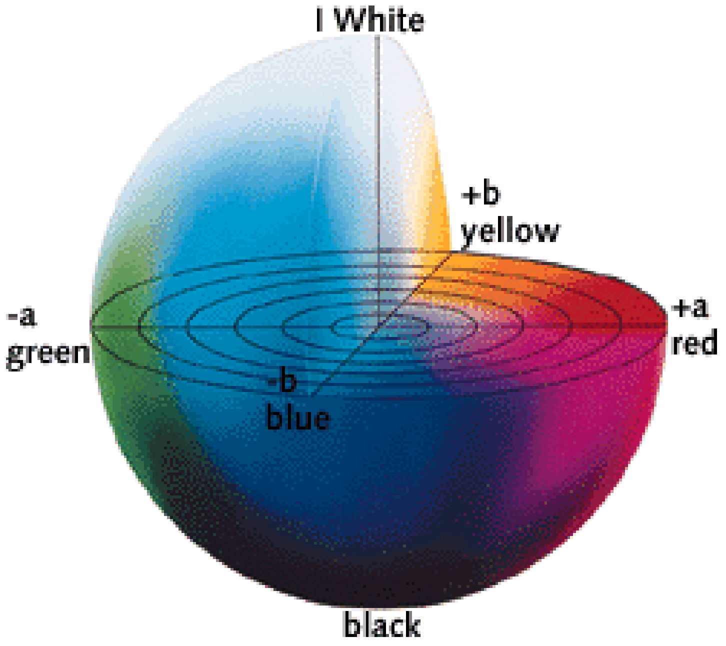
CIELAB, Lab, L*a*b

- One luminance channel (L)

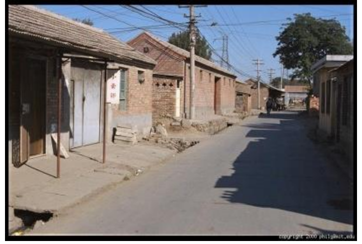
  and two color channels (a and b).

- In this model, the color differences which you perceive correspond to Euclidian distances in CIELab.

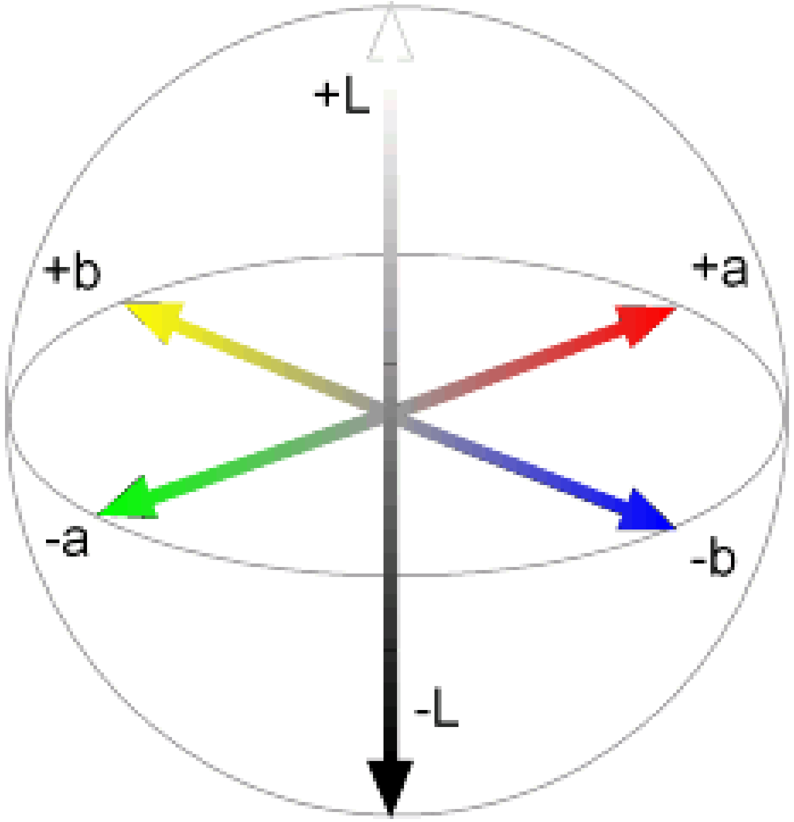
- The a axis extends from green (-a) to red (+a) and the b axis from blue (-b) to yellow (+b). The brightness (L) increases from the bottom to the top of the three-dimensional model.



Color spaces: L*a*b*



"Perceptually uniform"* color space



L
(a=0,b=0)

a
(L=65,b=0)

b
(L=65,a=0)

YIQ (for TV)

- The NTSC television standard is an encoding that uses one luminance value Y and two chromaticity values I and Q
- only luminance (i.e., Y) is used by black and white TVs, while all three (i.e., Y, I, and Q) are used by color TVs.
- the **Y** value is **encoded using more bits** than used for the values of I and Q because the human visual system is more sensitive to luminance (intensity) than to the chromaticity values.

An approximate transformation from RGB to YIQ:

$$Y = 0.30\ R + 0.59\ G + 0.11\ B$$

$$I = 0.60\ R - 0.28\ G - 0.32\ B$$

$$Q = 0.21\ R - 0.52\ G + 0.31\ B$$

YUV (JPEG and MPEG)

- YUV encoding is used in some digital video products and compression algorithms such as JPEG and MPEG.
- The conversion of RGB to YUV is as follows.

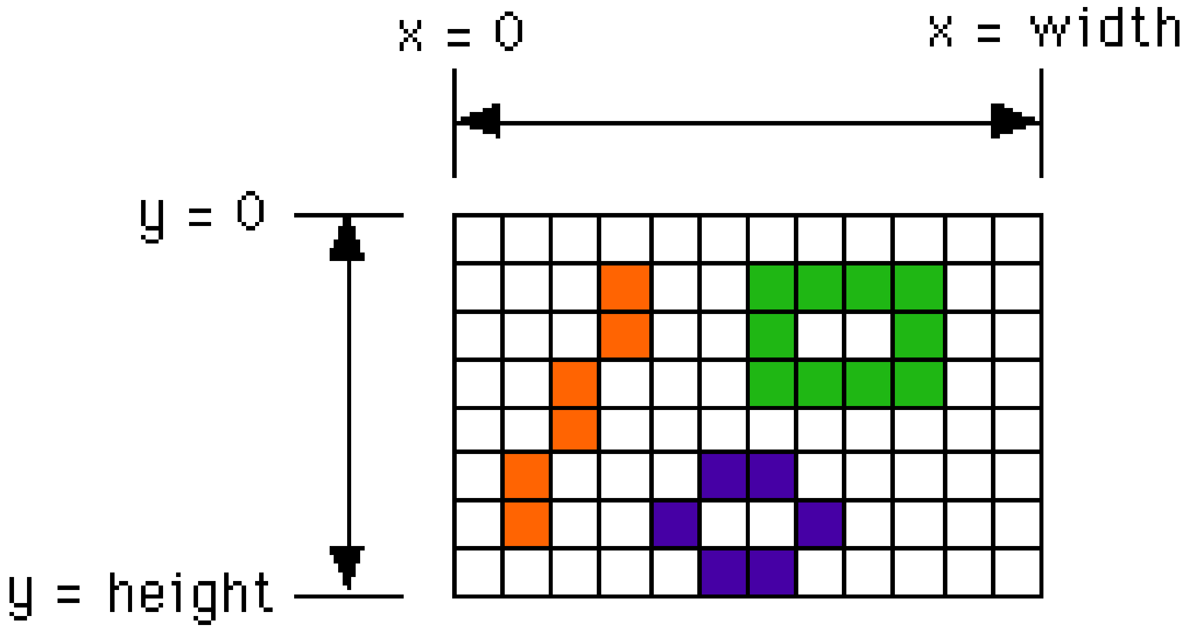
$$Y = 0.30R + 0.59G + 0.11B$$
$$U = 0.493 * (B - Y)$$
$$V = 0.877 * (R - Y)$$

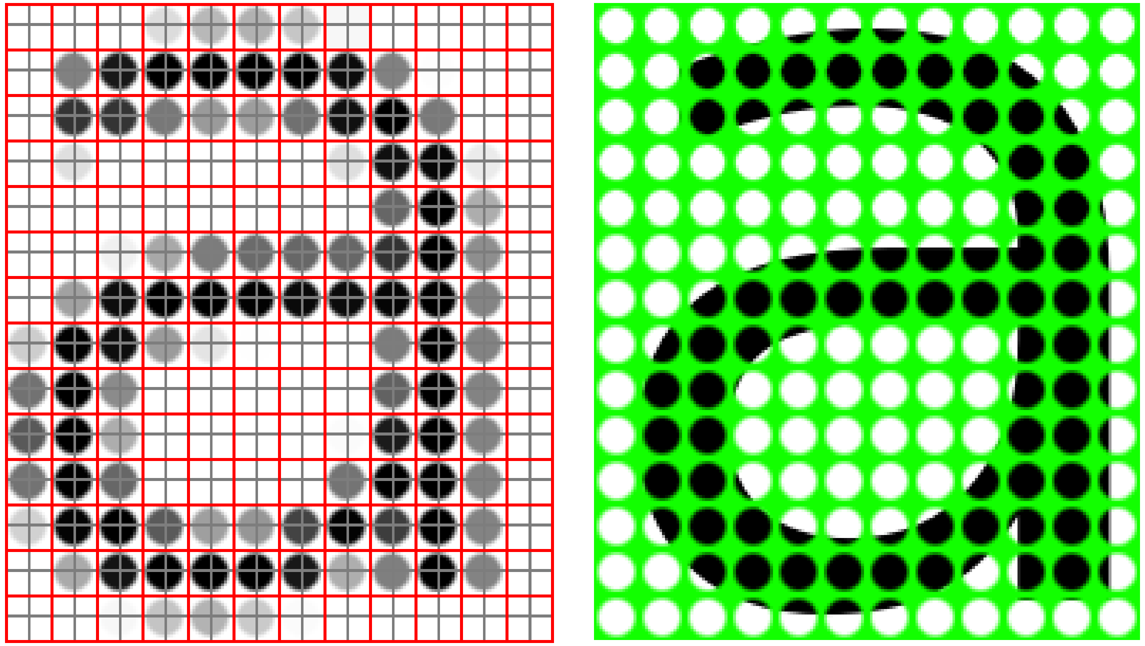
Color Spaces (summary)

• RGB	standard for cameras
• HSI/HSV	hue, saturation, intensity
• CIE L*a*b	intensity plus 2 color channels
• YIQ	color TVs, Y is intensity
• and more	

Still Images

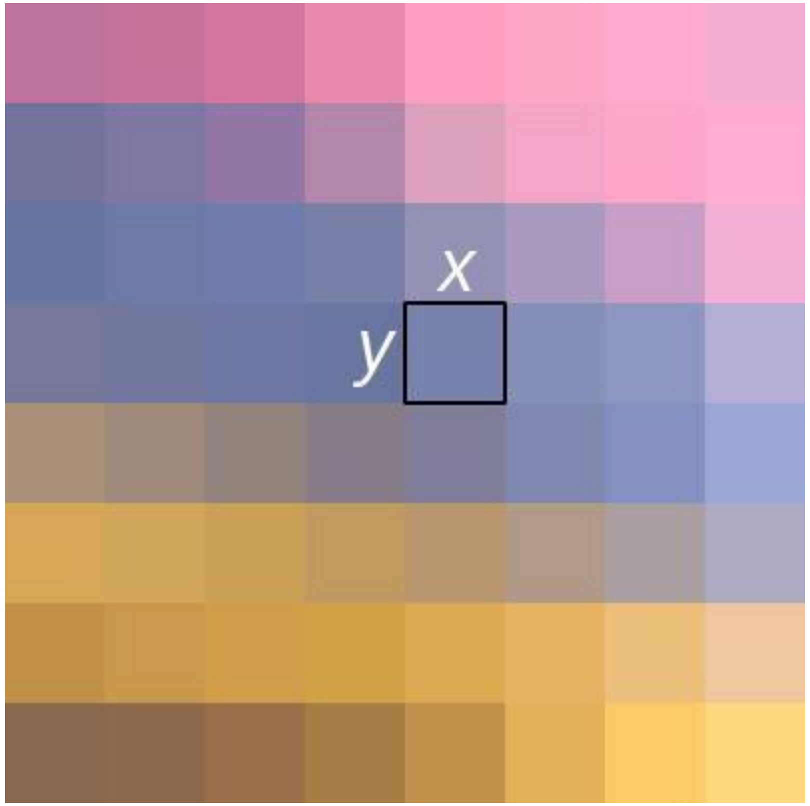
- **Pixels** (Picture Element) are **samples** of the original image
- Each pixel is expressed in a color space



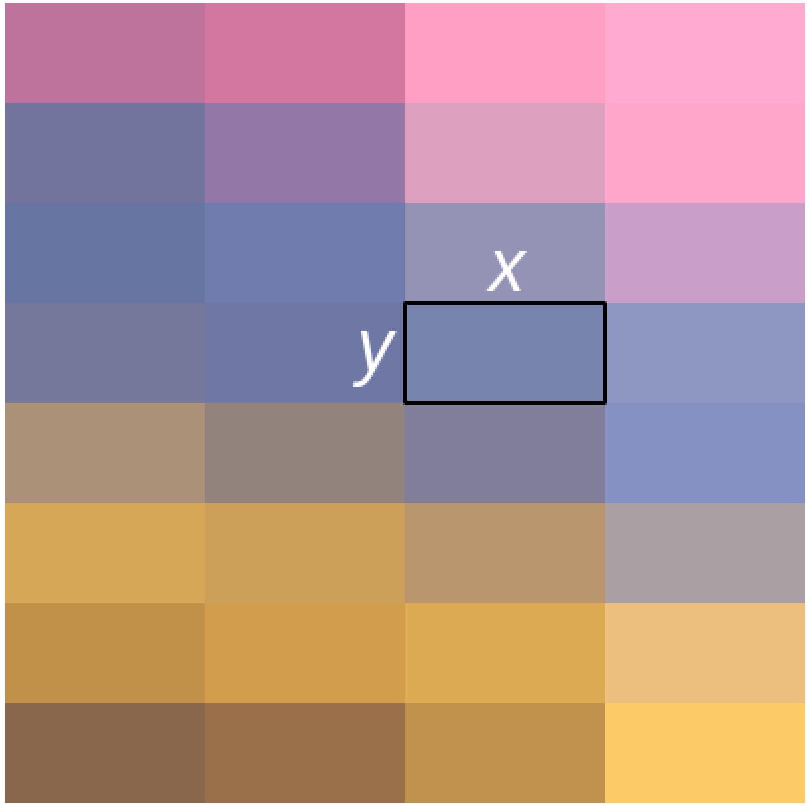


From the GIMP software documentation

- Describes how the width of a pixel compares to the height



1:1



2:1