Industrial Computer Vision

- Histogram & Filtering

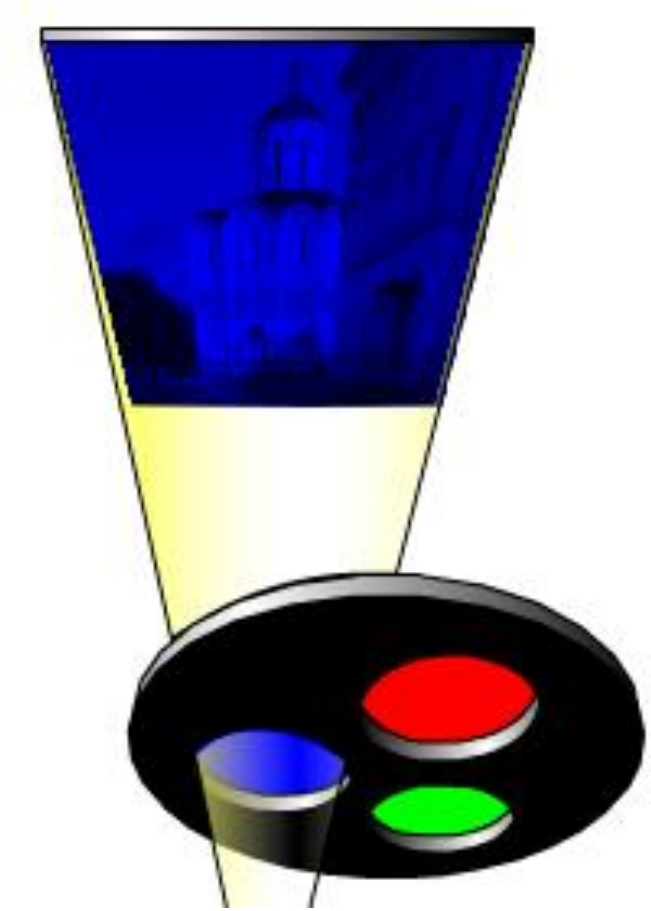
3rd lecture, 2022.09.21

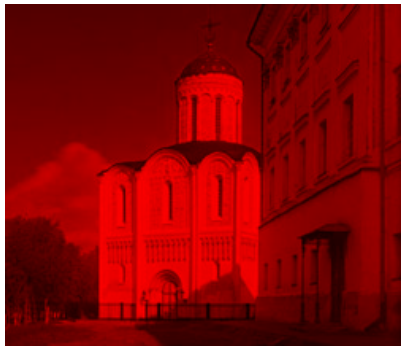
Lecturer: Youngbae Hwang

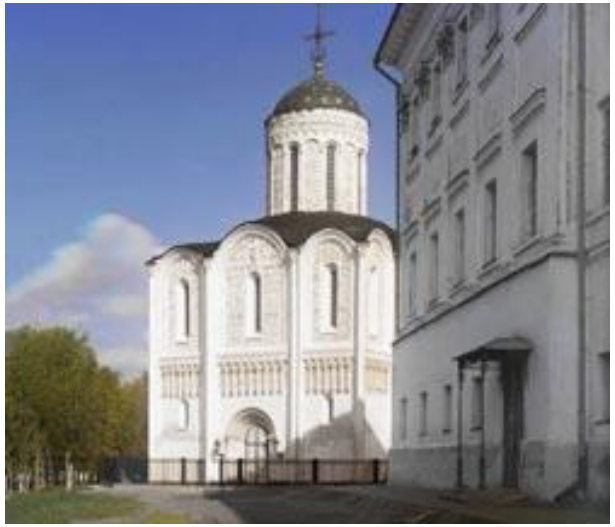
Contents

- Color image capture
- Color space conversion
- Gamma correction
- Histogram Equalization
- Image Filtering

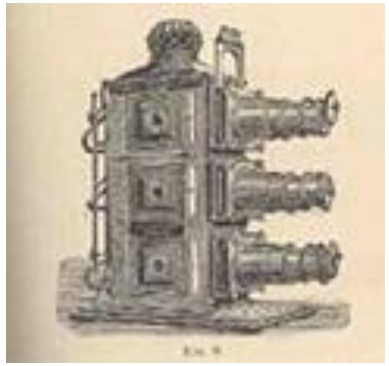
Color image capture







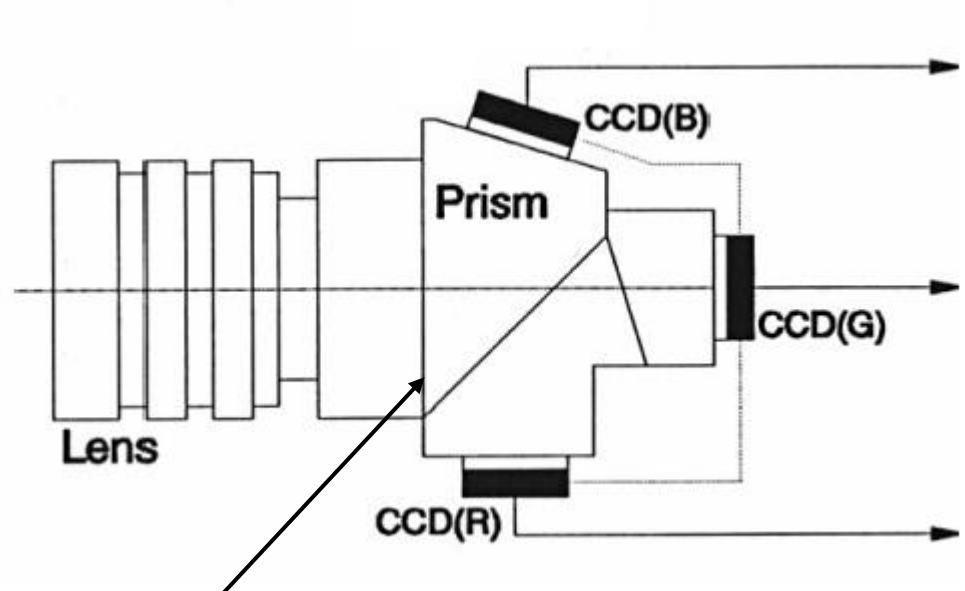
Prokudin-Gorskii (early 1990's)

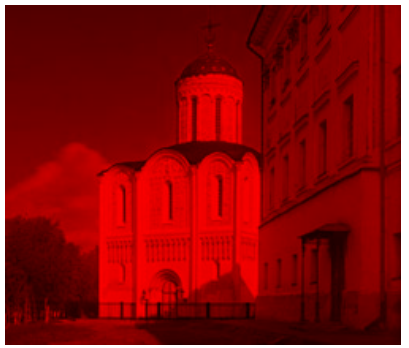


Lantern projector

Multi-chip for color imaging

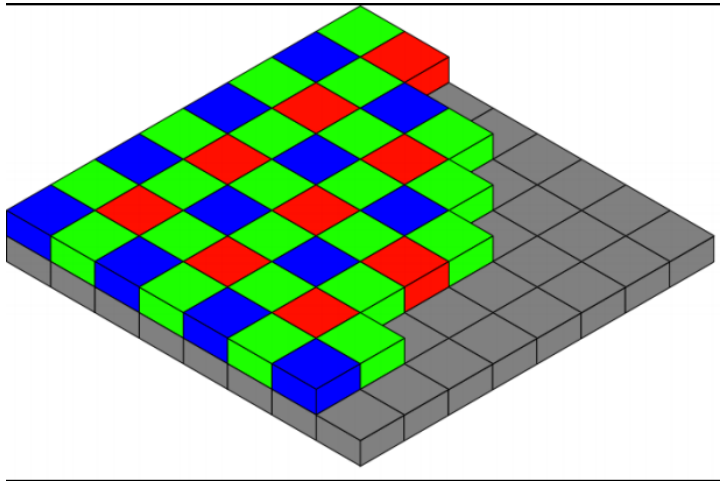
- Bulky and high cost

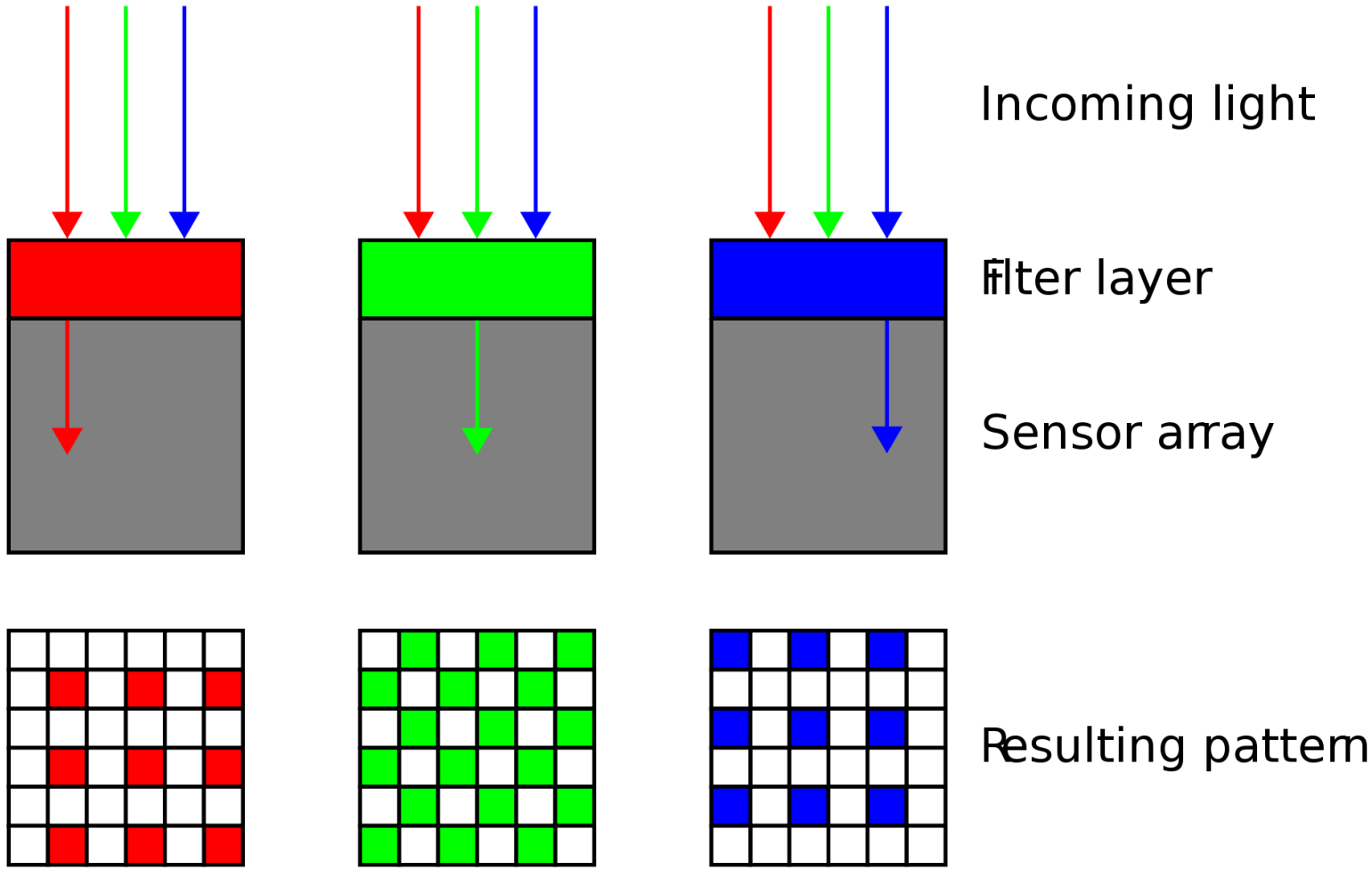




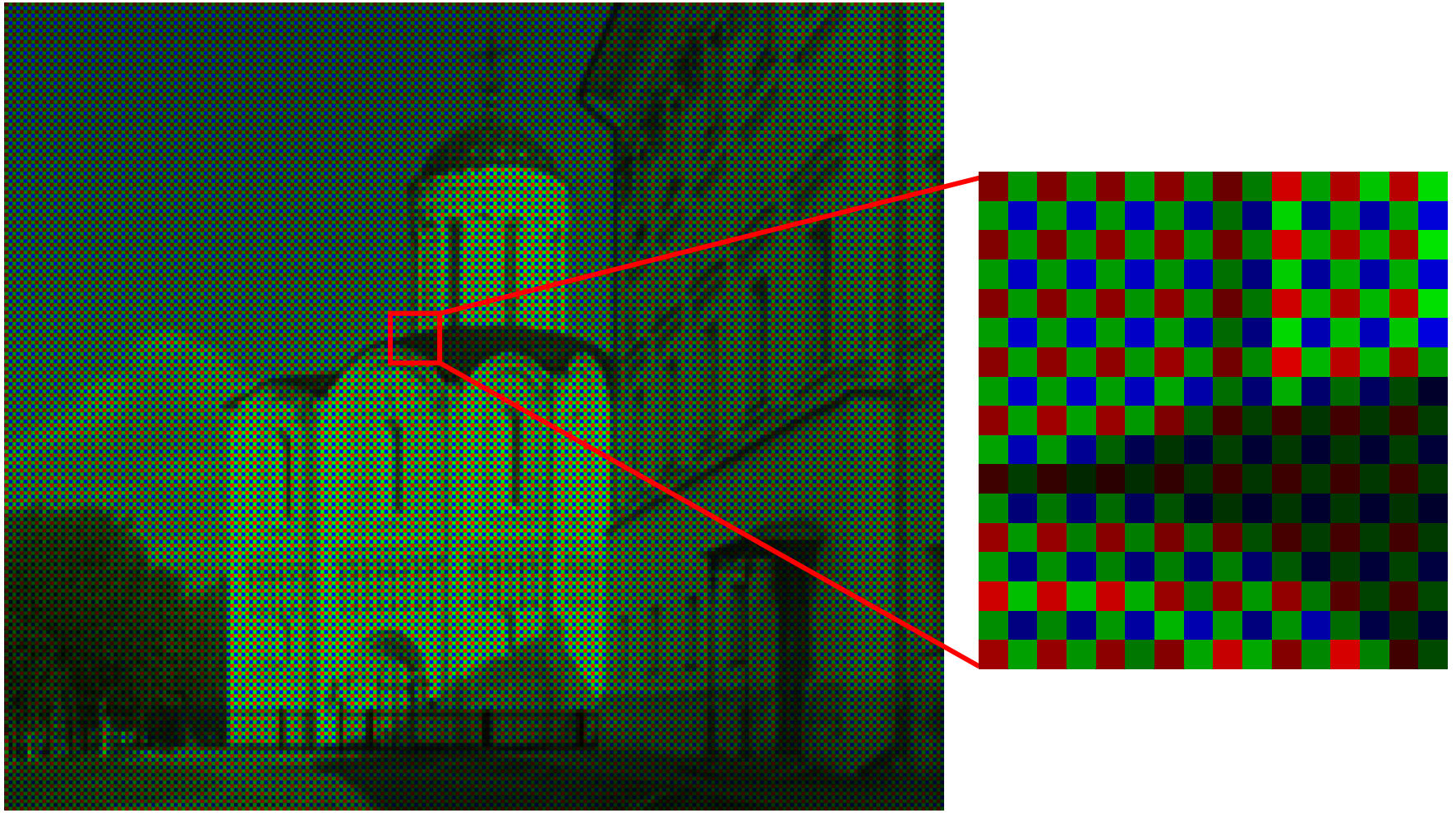
wavelength dependent

Bayer’s pattern for color imaging





Bayer’s pattern



Color Fundamentals

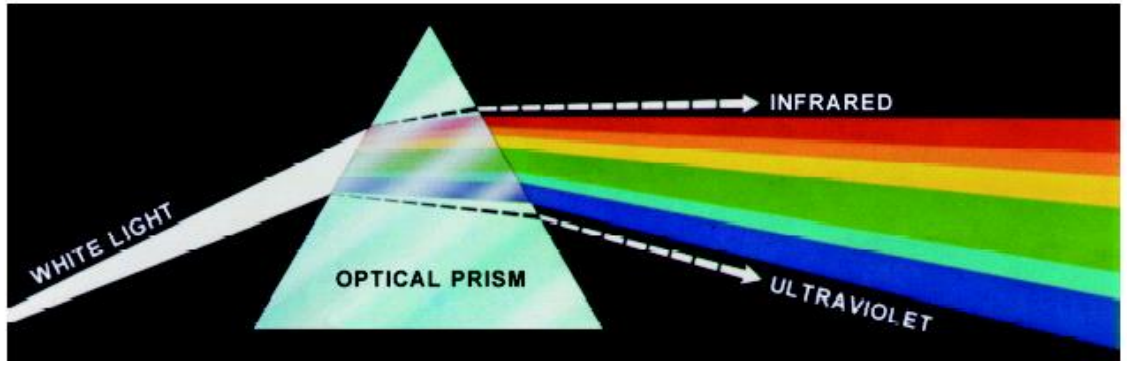


FIGURE 6.1 Color spectrum seen by passing white light through a prism. (Courtesy of the General Electric Co., Lamp Business Division.)

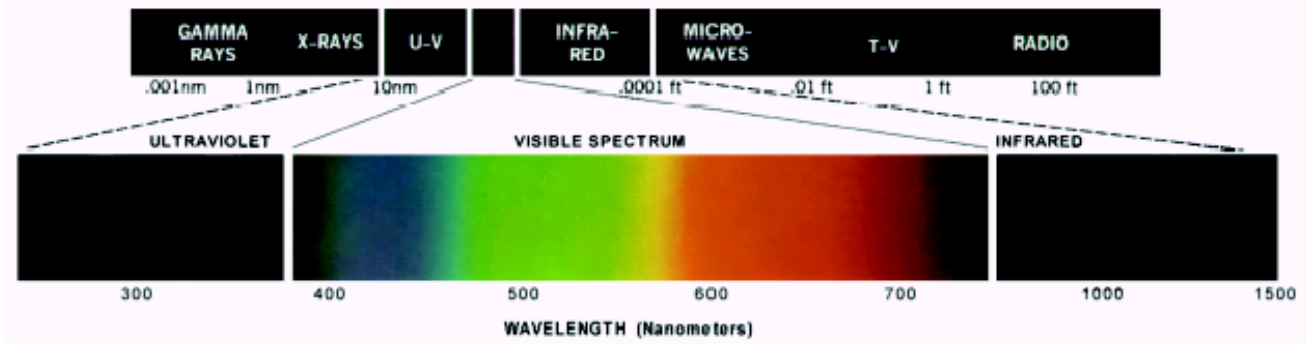


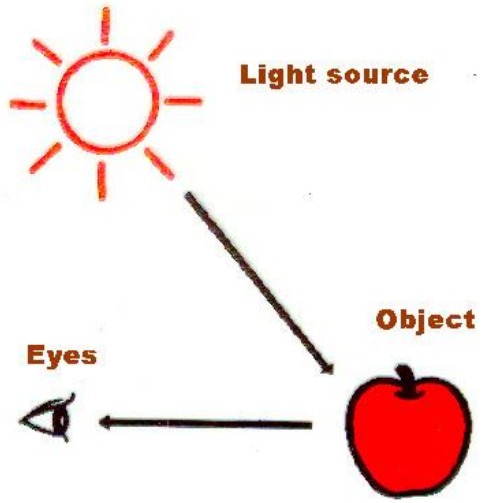
FIGURE 6.2 Wavelengths comprising the visible range of the electromagnetic spectrum. (Courtesy of the General Electric Co., Lamp Business Division.)

Color Fundamentals

- Visible light
  - Relatively, narrow band of frequencies in the electromagnetic energy spectrum
- Color perception
  - The colors that human beings perceive in an object are determined by the nature of the light reflected from the object
  - Some shades of color
    - A body that favors reflectance in a limited range of the visible spectrum
  - White
    - A body that reflects light that is balanced in all visible wavelengths

Color Fundamentals

- Color can only exist when three components are present:
  a viewer, an object, and light
- When white light hits an object, it selectively blocks some
  colors and reflects others
- Only the reflected colors contribute to the viewer's perception of color



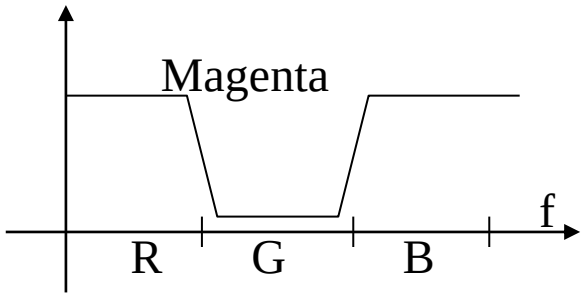
Color Fundamentals

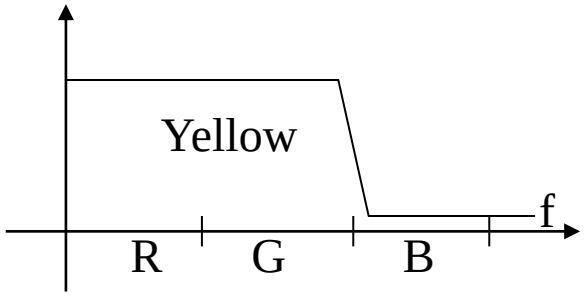
- Primary colors of light : red, green, blue
  - The CIE specific wavelength values to the three primary colors
    - Blue= 435.8 nm, green= 546.1 nm, red= 700 nm
  - No mean that these three fixed RGB components acting alone can generate all spectrum colors
- Secondary colors of light : cyan, magenta, yellow
  - Magenta: R+B, cyan: G+B, yellow: R+G
  - Mixing the three primaries, or a secondary with its opposite primary color, in the right intensities produces white light

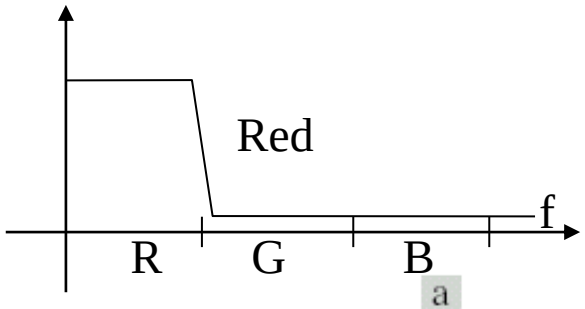
Color Fundamentals

- Primary colors of pigments or colorants
  - cyan, magenta, yellow
  - A primary color of pigments is defined as one that subtracts or absorbs a primary color of light and reflects or transmits the other two
- Secondary colors of pigments or colorants
  - red, green, blue
  - Combination of the three pigment primaries, or a secondary with its opposite primary, produces black

Color Fundamentals

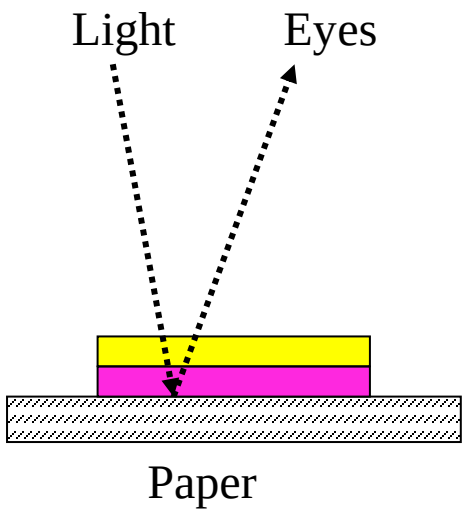






a
b

$$f(x) = i(x)\ r(x)$$



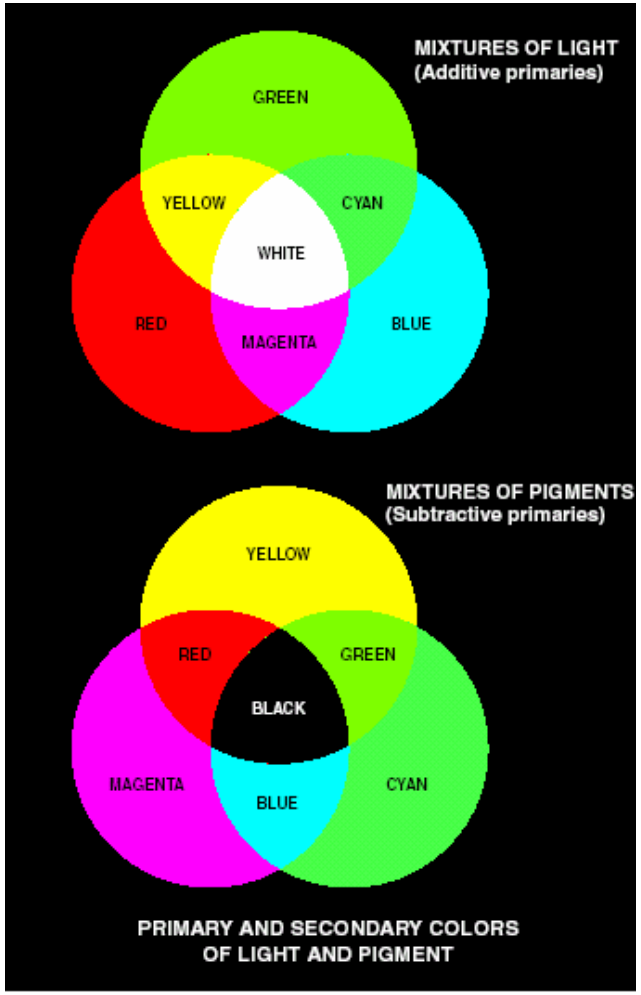


FIGURE 6.4 Primary and secondary colors of light and pigments. (Courtesy of the General Electric Co., Lamp Business Division.)

Color Fundamentals

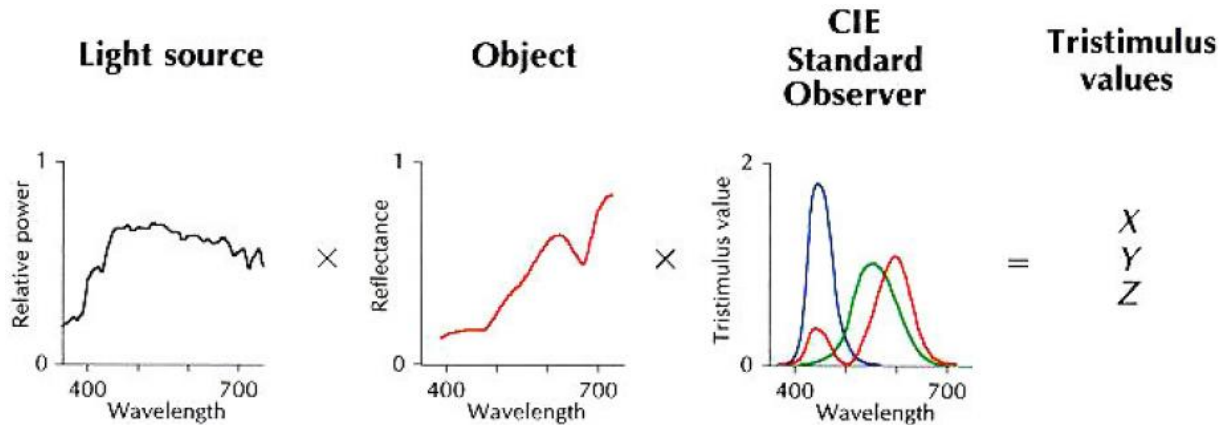
- Characteristics of colors
  - Brightness
    - The chromatic notion of intensity
  - Hue
    - An attribute associated with the dominant wavelength in a mixture of light waves
    - Representing dominant color as perceived by an observer
  - Saturation
    - Referring to relative purity or the amount of white mixed with a hue
    - Saturation is inversely proportional to the amount of white light

Color Fundamentals

- Hue and saturation taken together are called chromaticity
  - A color may be characterized by its brightness and chromaticity
  - The amounts of red, green, and blue needed to form any particular color are called the tristimulus values
    - Denoted X(red), Y(green), and Z(blue)

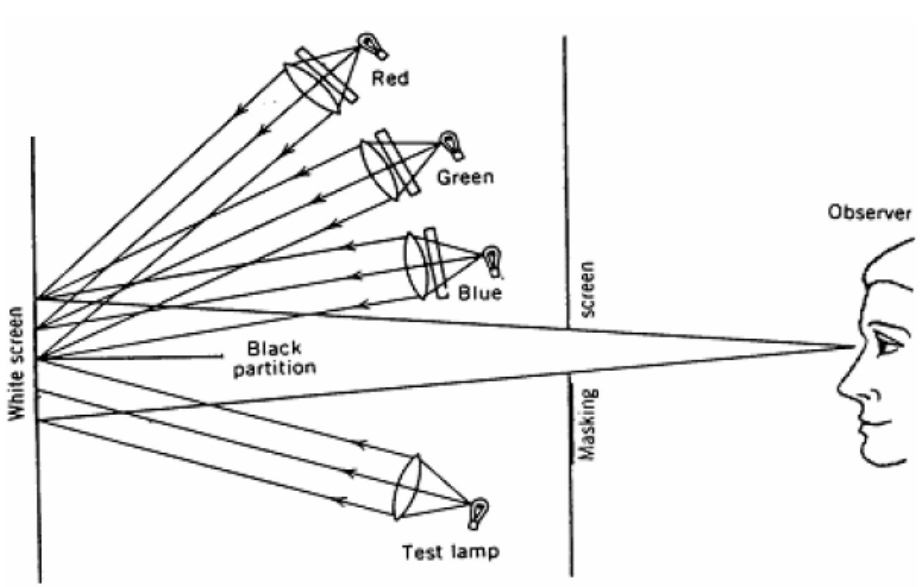
$$x = \frac{X}{X+Y+Z}, \quad y = \frac{Y}{X+Y+Z}, \quad z = \frac{Z}{X+Y+Z}$$

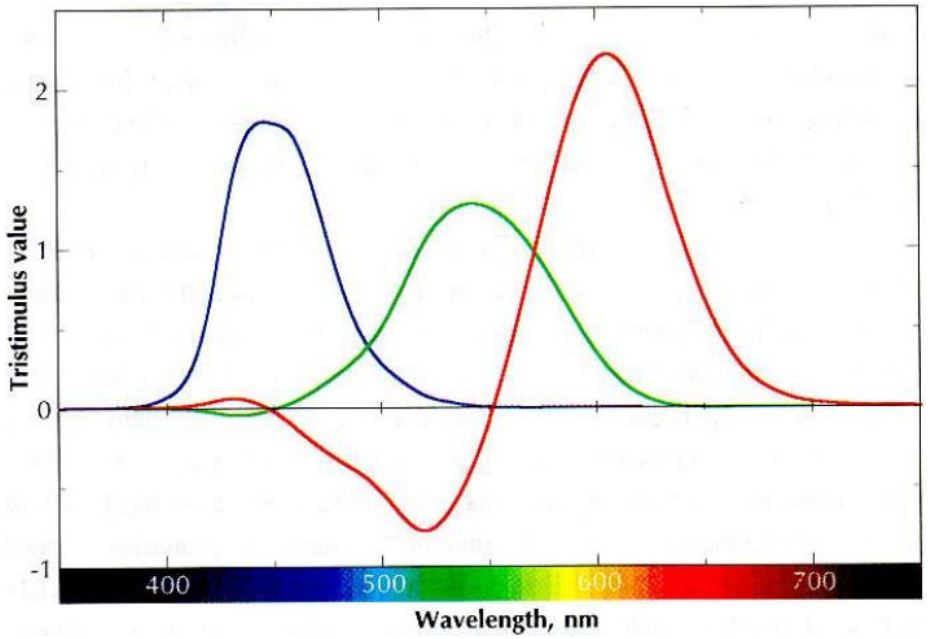
$$x + y + z = 1 \ , \ X = \frac{x}{y}Y, Z = \frac{z}{y}Y$$



Color Fundamentals

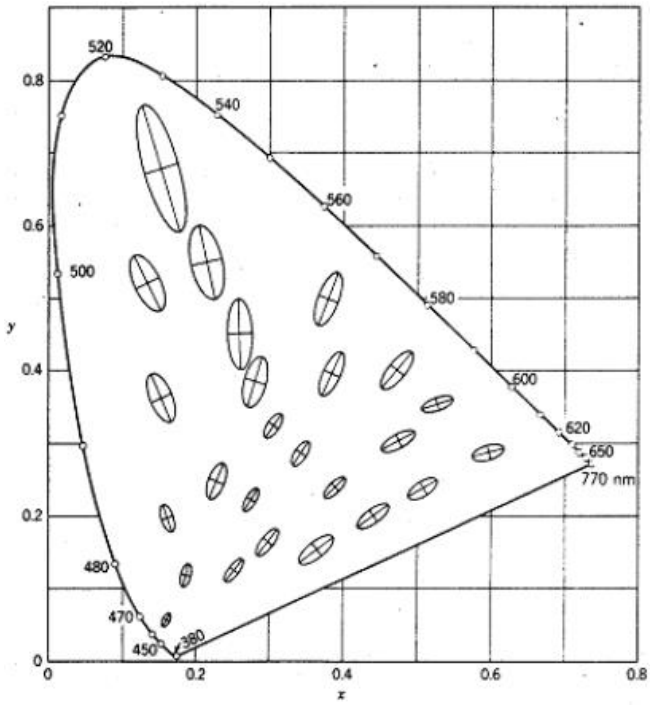
- CIE 1931 RGB Color matching function
  - the entire range of human color perception could be covered
  - 435.8nm(B), 546.1nm(G), 700nm(R)
  - 2°observer : the color-sensitive cones resided within a 2° arc of the fovea
  - There is the minus sign : Need 6-ch

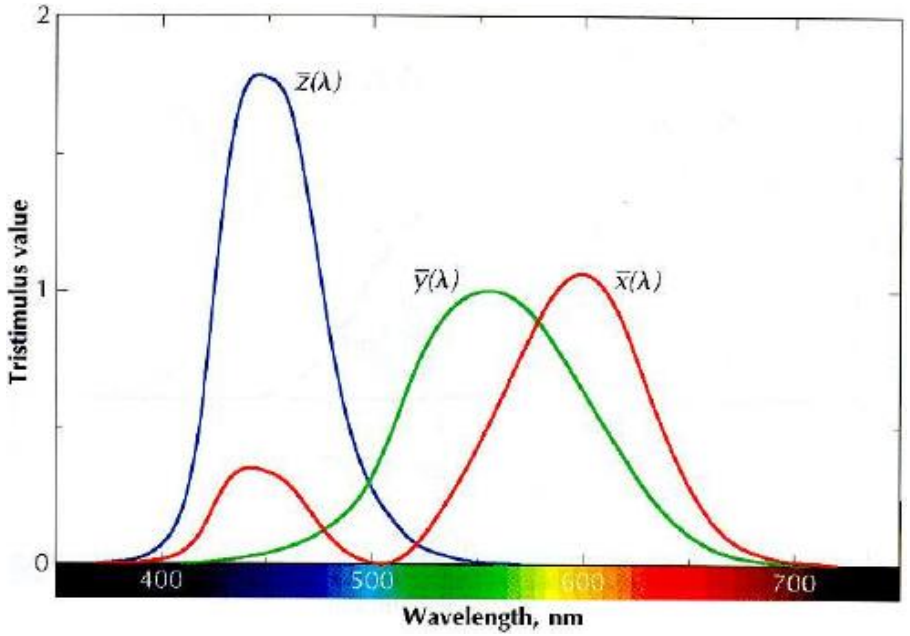




Color Fundamentals

- CIE 1931 XYZ Color matching function
  - Overcomes minus sign problem.
  - 2°observer
  - Perceptual non-uniformity





Color Fundamentals

- Chromaticity diagram
  - Showing color composition as a function of x(R) and y(G)
  - For any value of x and y, z(B) = 1 – (x + y)
  - The positions of the various spectrum colors are indicated around the boundary of the tongue-shaped
  - The point of equal energy = equal fractions of the three primary colors = CIE standard for white light
  - Boundary : completely saturated
  - Useful for color mixing
  - A triangle(RGB) does not enclose the entire color region

Color Fundamentals

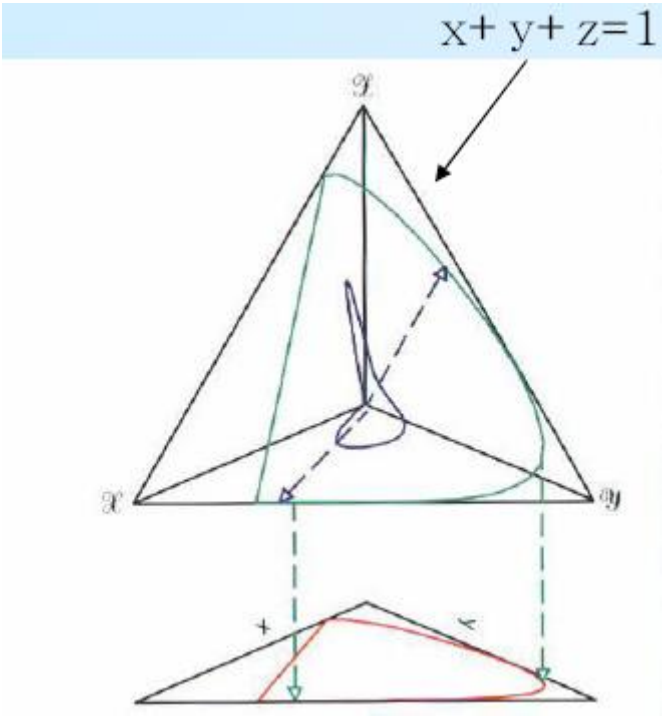
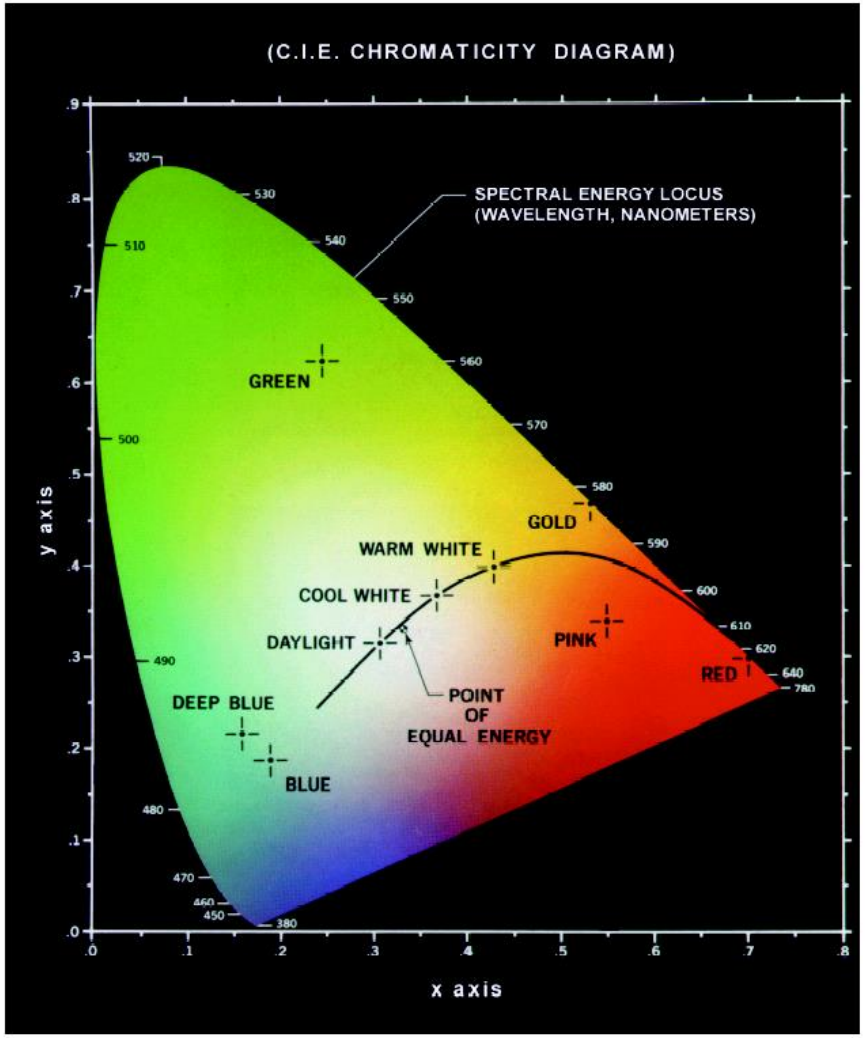
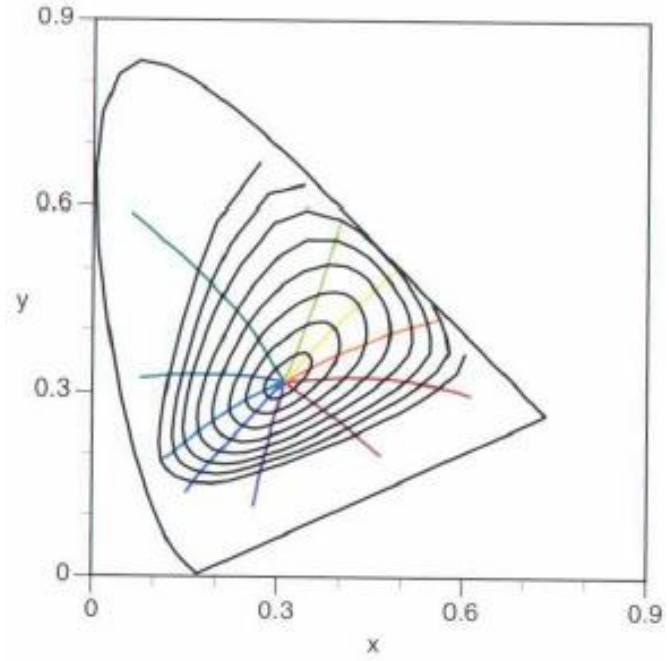


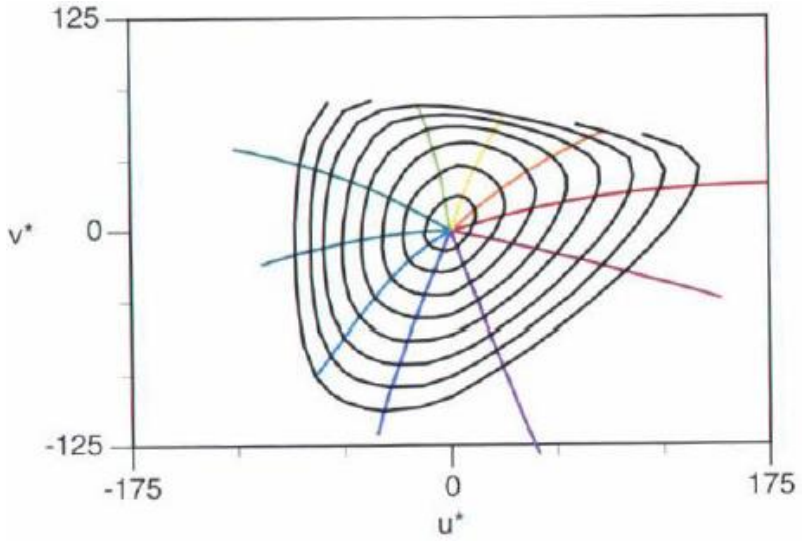
FIGURE 6.5 Chromaticity diagram. (Courtesy of the General Electric Co., Lamp Business Division.)

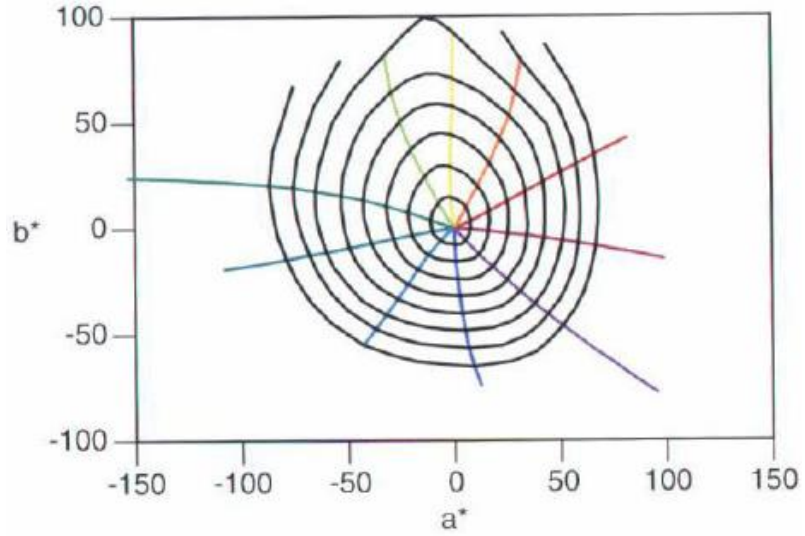


Color Fundamentals

- Perceptual uniformity
  - CIE 1976 L*a*b*
  - CIE 1976 L*u*v*





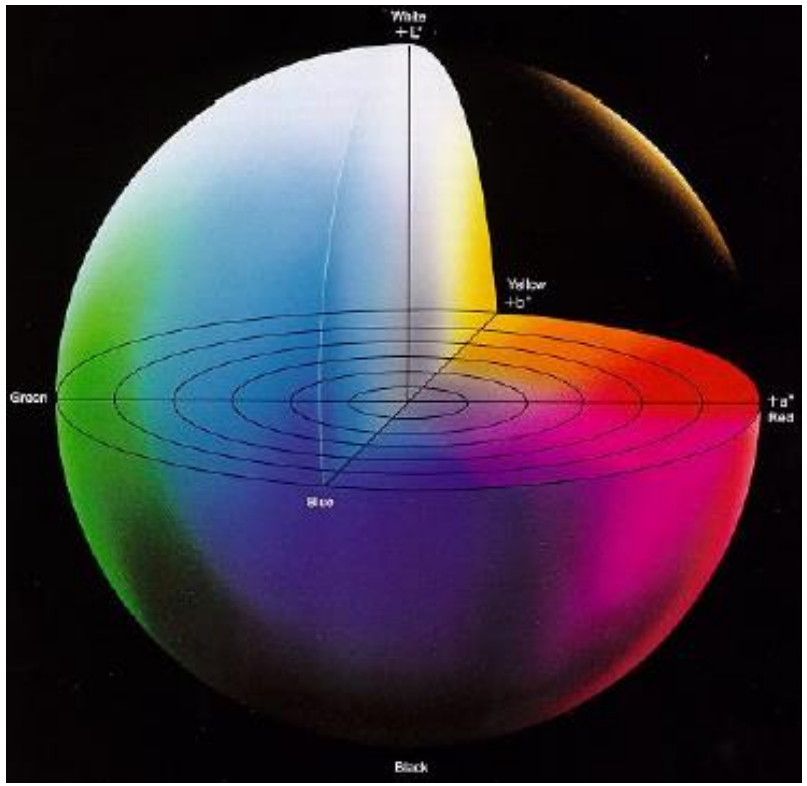


Tone and Color Corrections

- CIE 1976 L*a*b*
  - Lab color is designed to approximate human vision
  - Uniform color space
  - Correspond to the three characteristics of color
  - It is an excellent decoupler of intensity and color.
    - It is useful for making in both image manipulation and image compression
  - Its gamut encompasses the entire visible spectrum
  - It can represent accurately the colors of any device
    - Display, print or scanner etc.
  - Device independent color model=>Color management systems

Tone and Color Corrections

- CIE 1976 L*a*b*



$$L^* = 116 \cdot h\left(\frac{Y}{Y_W}\right) - 16$$

$$a^* = 500\left[h\left(\frac{X}{X_W}\right) - h\left(\frac{Y}{Y_W}\right)\right]$$

$$b^* = 200\left[h\left(\frac{Y}{Y_W}\right) - h\left(\frac{Z}{Z_W}\right)\right]$$

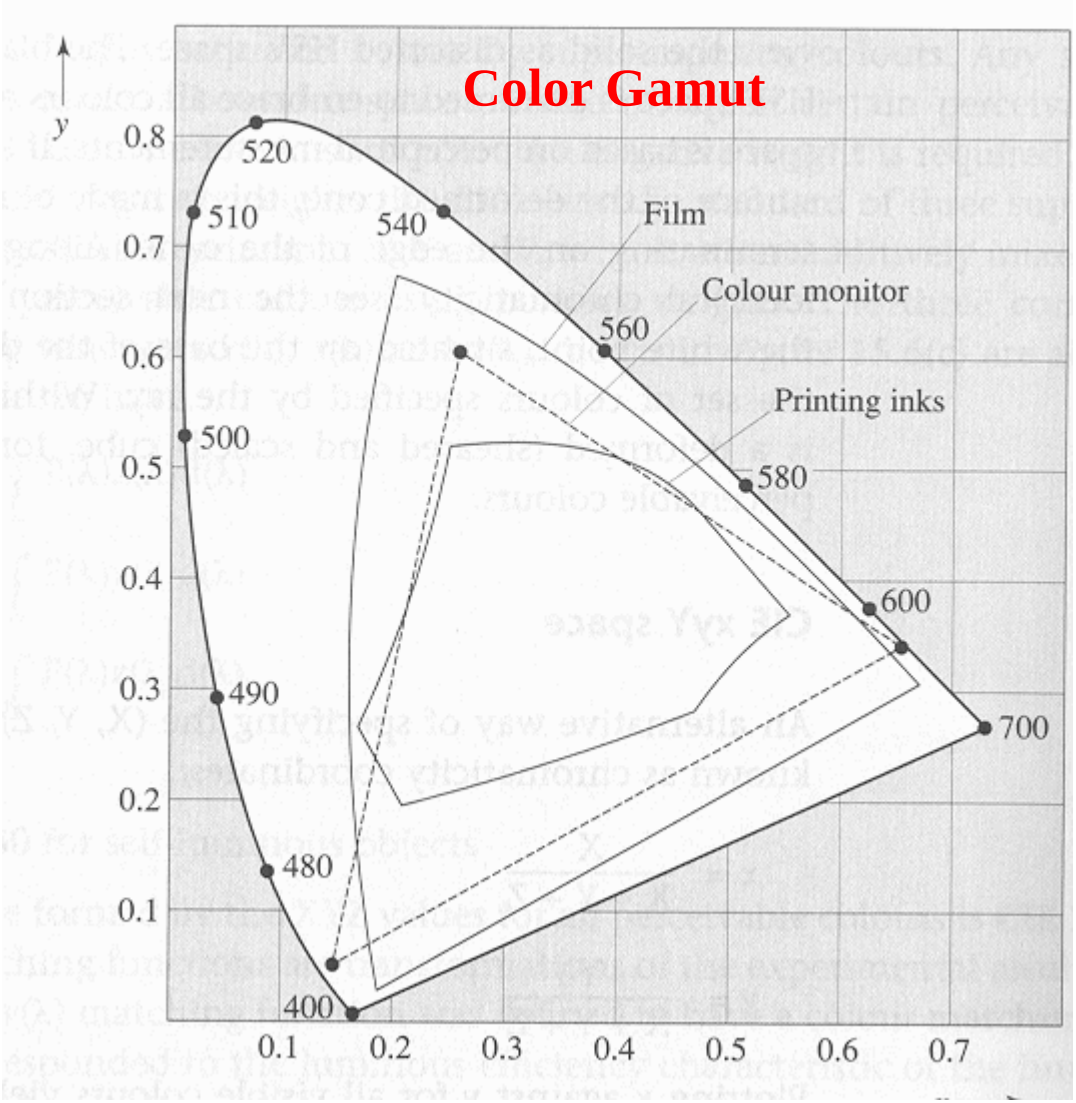
$$h(q) = \sqrt[3]{q} \qquad (q > 0.008856)$$

$$= 7.787q + 16/116 \quad (q \le 0.008856)$$

D65 standard illumination	
x	0.3127
y	0.329
Xw	95.04559271
Yw	100
Zw	108.9057751

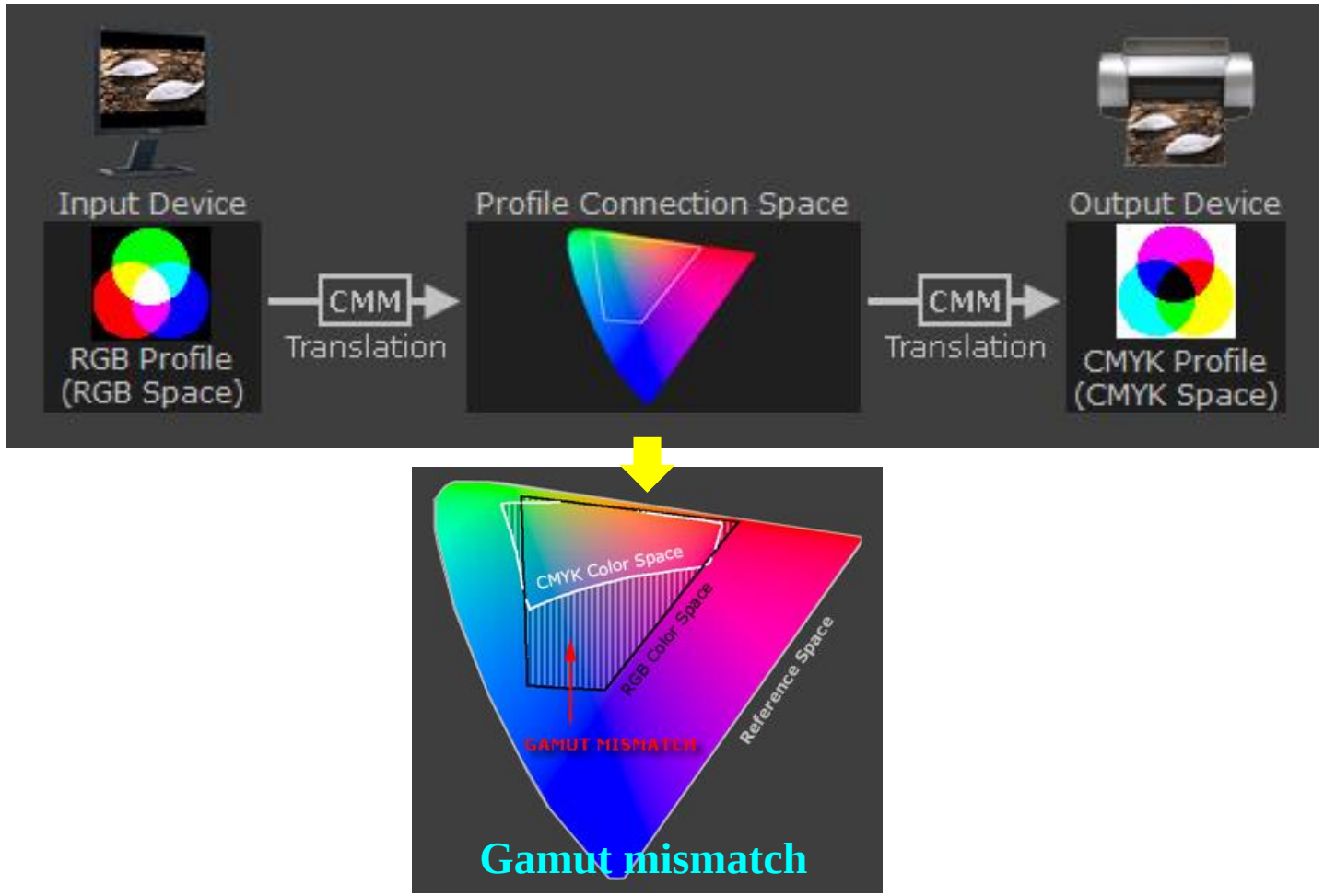
Color Fundamentals

- The range of color representation of a **display device**



Color Models

- **Color space conversion** translating color from one device's space to another

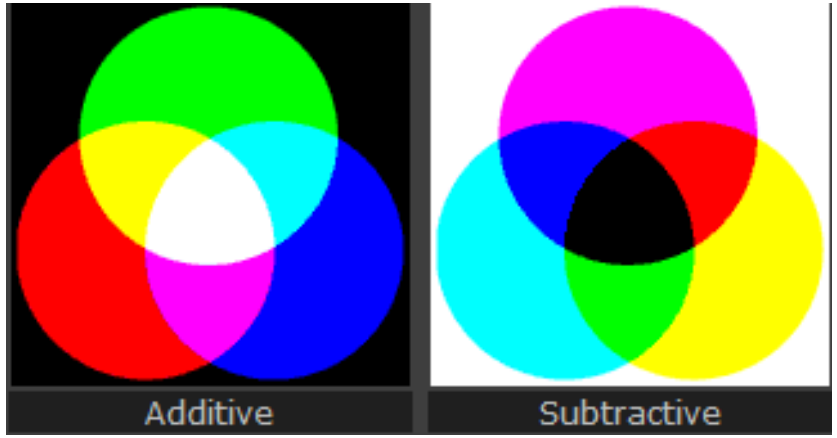


Color Models

- The purpose of a color model
  - To facilitate the specification of colors in some standard
- Color models are oriented either toward hardware or applications
  - Hardware-oriented
    - Color monitor or Video camera : RGB
    - Color printer : CMY
    - Color TV broadcast : YIQ (I : inphase, q : quadrature)
  - Color image manipulation : HSI, HSV
  - Image processing : RGB, YIQ, HSI

Color Models

- Additive processes create color by adding light to a dark background **(Monitors)**
- Subtractive processes use pigments or dyes to selectively block white light **(Printers)**



Additive Color Mixing			Subtractive Color Mixing		
Red + Green	→	Yellow	Cyan + Magenta	→	Blue
Green + Blue	→	Cyan	Magenta + Yellow	→	Red
Blue + Red	→	Magenta	Yellow + Cyan	→	Green
Red + Green + Blue	→	White	Cyan + Magenta + Yellow	→	Black

The RGB Color Model

- RGB model is based on a Cartesian coordinate system
  - The color subspace of interest is the cube
  - RGB values are at three corners
  - Colors are defined by vectors extending from the origin
  - For convenience, all color values have been normalized
    - All values of R, G, and B are in the range [0, 1]

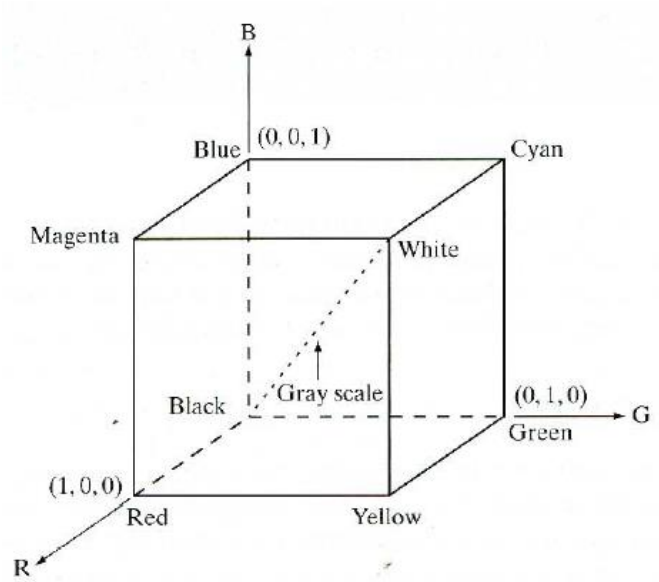


FIGURE 6.7
Schematic of the RGB color cube. Points along the main diagonal have gray values, from black at the origin to white at point (1, 1, 1).

The RGB Color Model

- Images represented in the RGB color model
  - Images represented in the RGB color model consist of three independent image planes, one for each primary color
- Pixel depth
  - The number of bits used to represent each pixel in RGB space is called the pixel depth
  - The term full-color image is used often to denote 24-bit RGB color image

The RGB Color Model

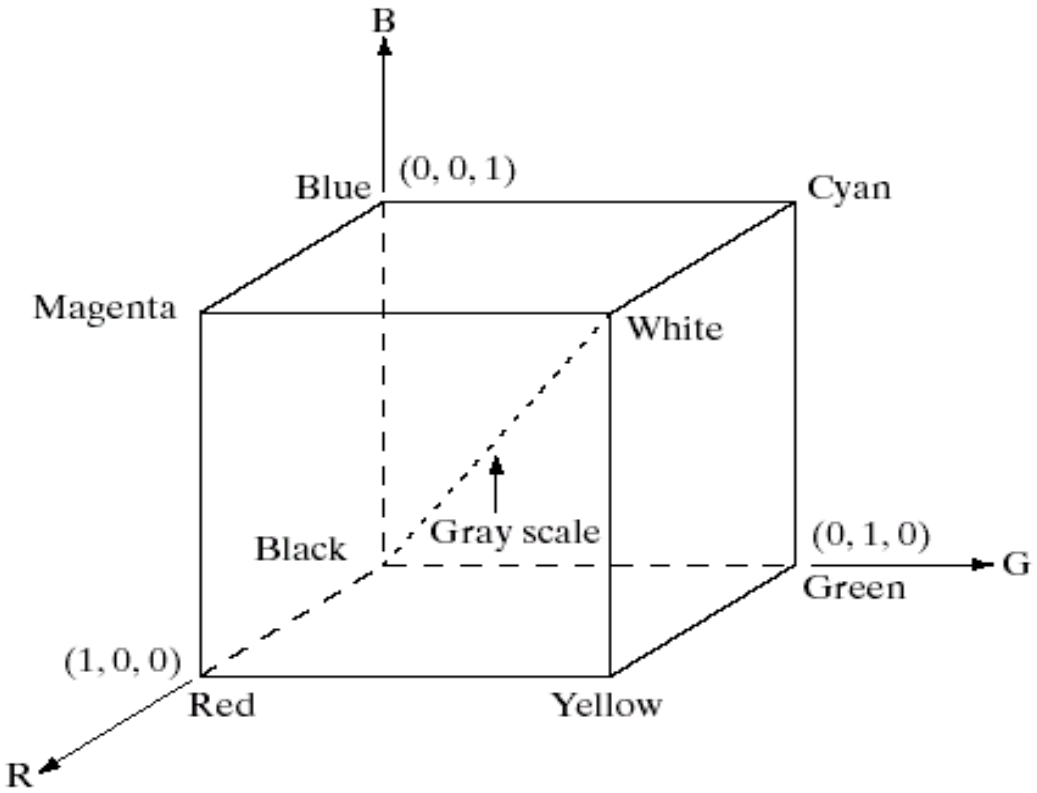


FIGURE 6.7
Schematic of the RGB color cube. Points along the main diagonal have gray values, from black at the origin to white at point (1, 1, 1).

The RGB Color Model

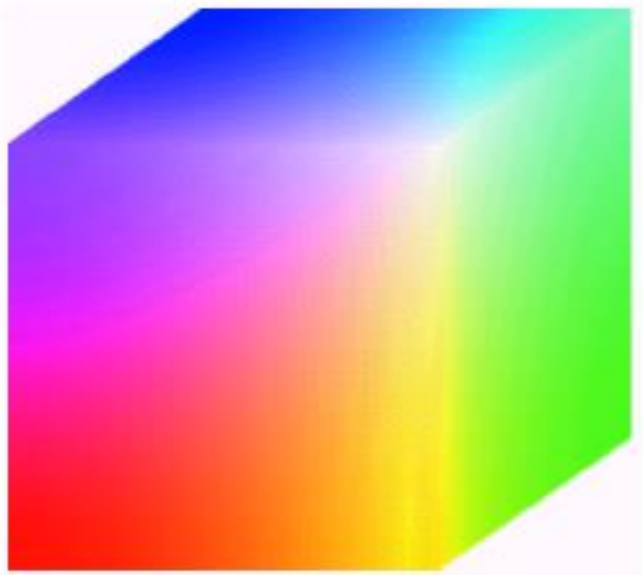


FIGURE 6.8 RGB 24-bit color cube.

The CMY and CMYK Color Models

- General purpose of CMY color model
  - To generate hardcopy output
- The primary colors of pigments
  - Cyan, magenta, and yellow
  - C = W - R, M = W - G, and Y = W - B
- Most devices that deposit colored pigments on paper require CMY data input
  - Converting RGB to CMY

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

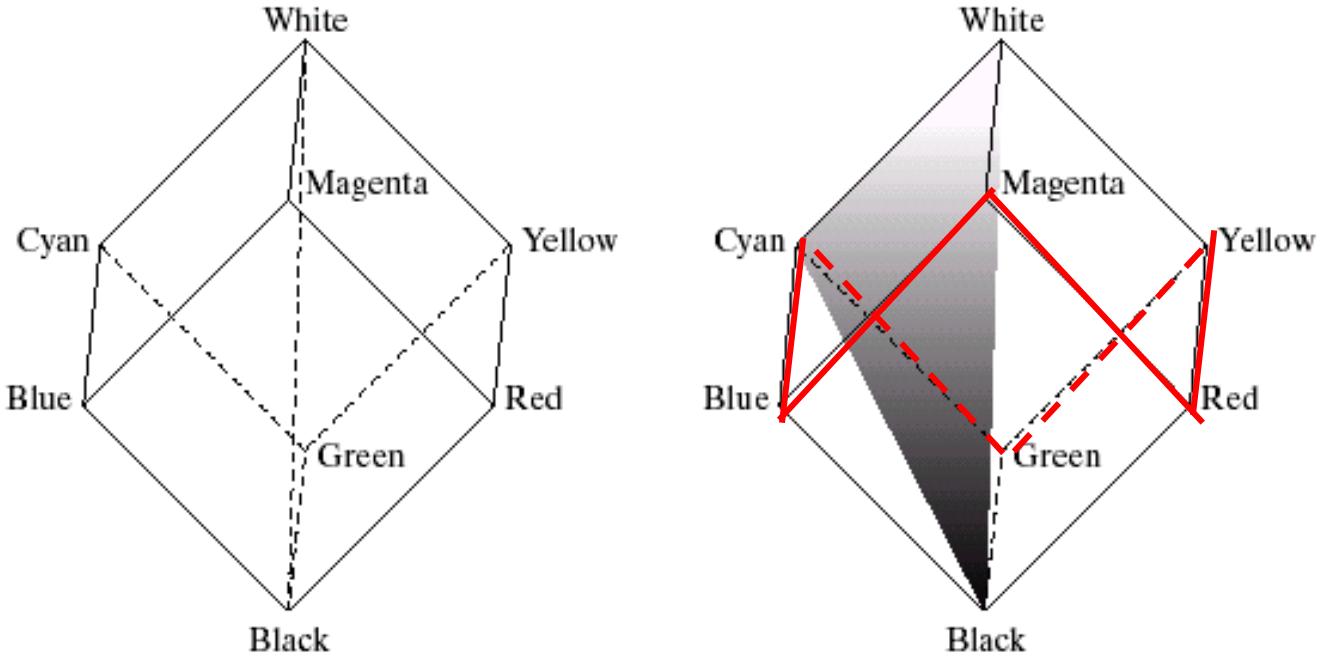
  - The inverse operation from CMY to RGB is generally of no practical interest

The HSI Color Model

- Usefulness
  - The intensity is decoupled from the color information
  - The hue and saturation are intimately related to the way in which human beings perceive color
  - An ideal tool for developing image procession algorithms based on some of the color sensing properties of the human visual system

The HSI Color Model

- HSI color model
  - H : hue
  - S : saturation
  - I : intensity

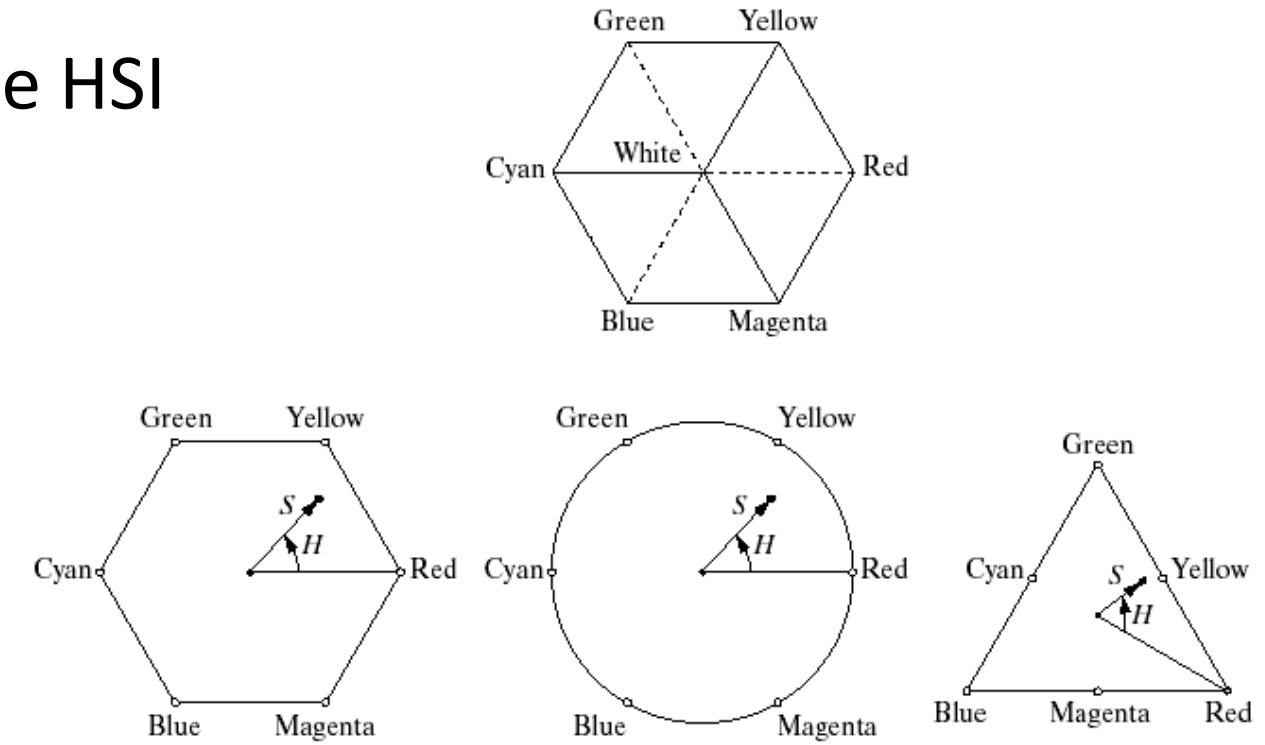


a b

FIGURE 6.12 Conceptual relationships between the RGB and HSI color models.

The HSI Color Model

- Hue and saturation in the HSI color model
  - Hue - Angle from the red axis
  - Saturation - Length of the vector

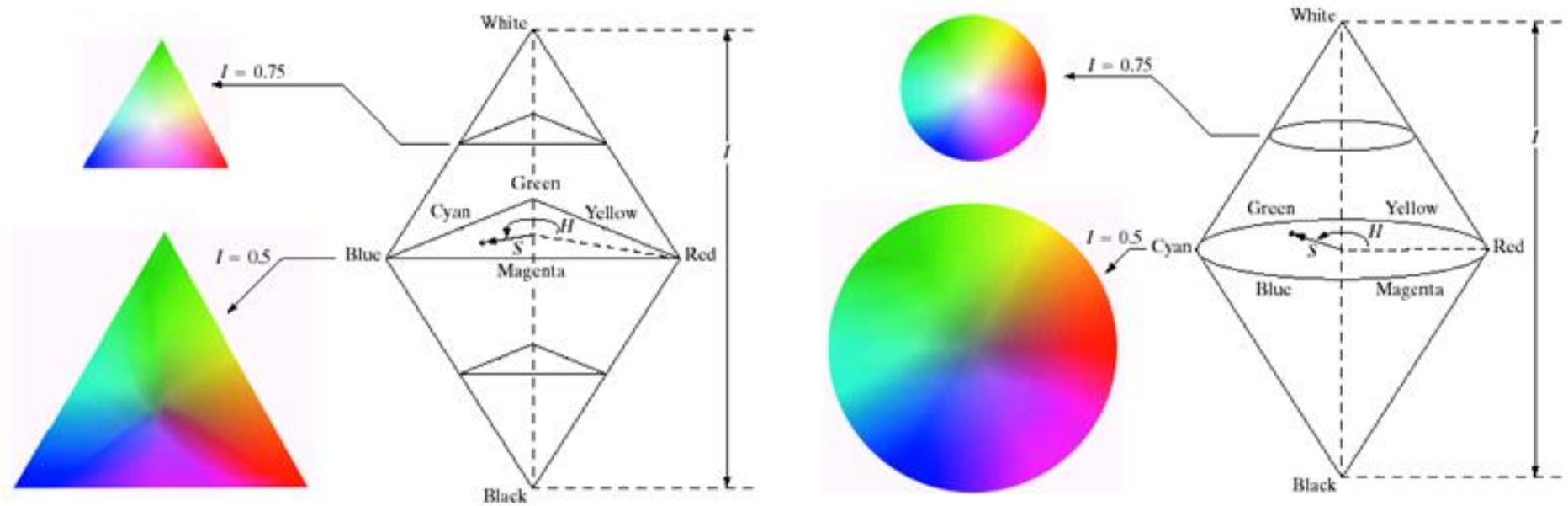


a
b c d

FIGURE 6.13 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.

The HSI Color Model

- The HSI color model based on triangular and circular color planes



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.

The HSI Color Model

- Converting colors from RGB to HIS
  - Hue component

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

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

  - Saturation component

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

  - Intensity component

$$I = \frac{1}{3}(R+G+B)$$

RGB values have been normalized to the range [0,1]

Angle θ is measured with respect to the red axis

Hue can be normalized to the range [0, 1] by dividing by 360°c

The HSI Color Model

- Converting colors from HSI to RGB
  - Three sectors of interest, corresponding to the 120^0 intervals in the separation of primaries
  - RG sector

$$(0^\circ \le H < 120^\circ)$$

$$B = I(1-S)$$

$$R = I\left[1+\frac{S\cos H}{\cos(60^o - H)}\right]$$

$$G = 3I-(R+B)$$

The HSI Color Model

Converting colors from HSI to RGB(continued)

- GB sector $(120^\circ \le H < 240^\circ)$

$$H = H - 120^o$$

$$R = I(1-S)$$

$$G = I\left[1 + \frac{S\cos H}{\cos(60^o - H)}\right]$$

$$B = 3I - (R+G)$$

- BR sector $(240^\circ \le H < 360^\circ)$

$$H = H - 240^o$$

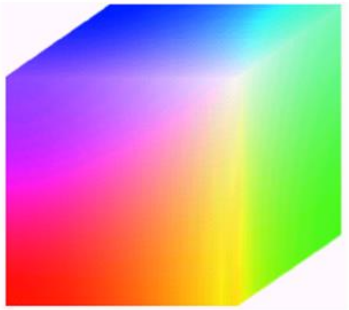
$$G = I(1-S)$$

$$B = I\left[1 + \frac{S\cos H}{\cos(60^o - H)}\right]$$

$$R = 3I - (G+B)$$

Converting colors from HSI to RGB

- Example
  - The HSI values corresponding to the image of the RGB color cube



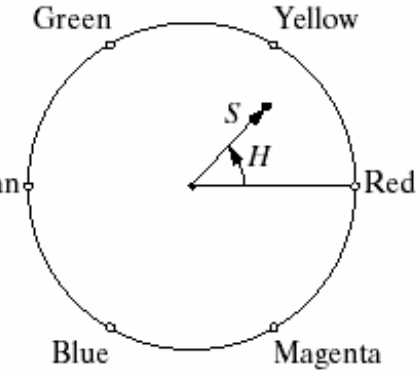
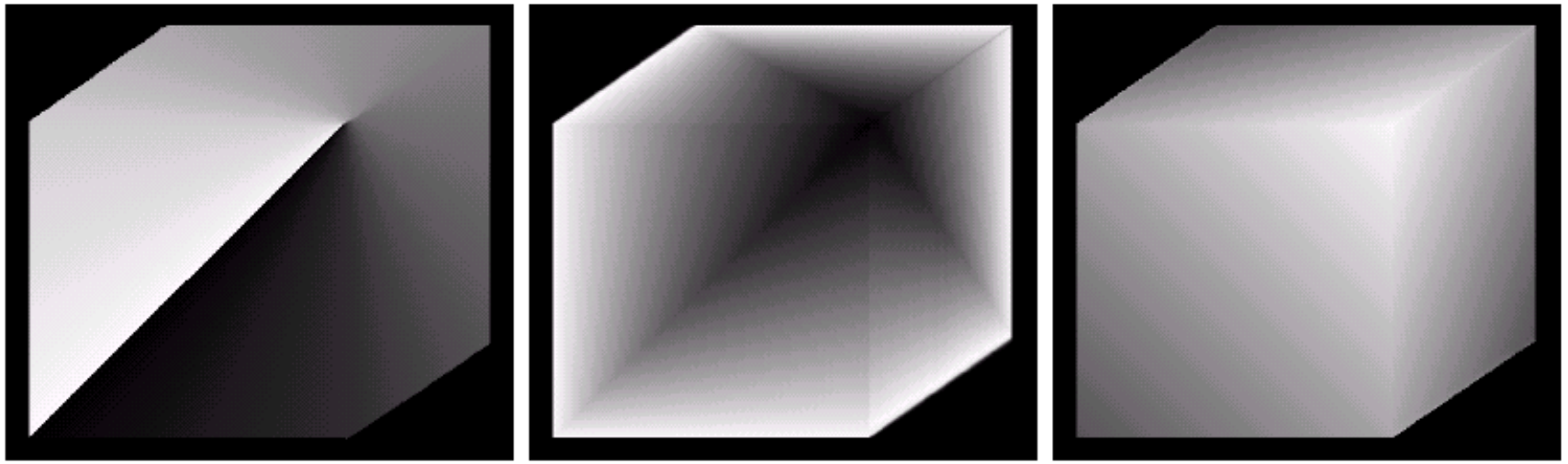


FIGURE 6.8

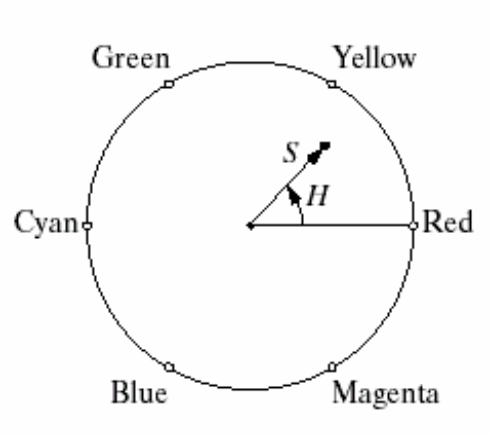


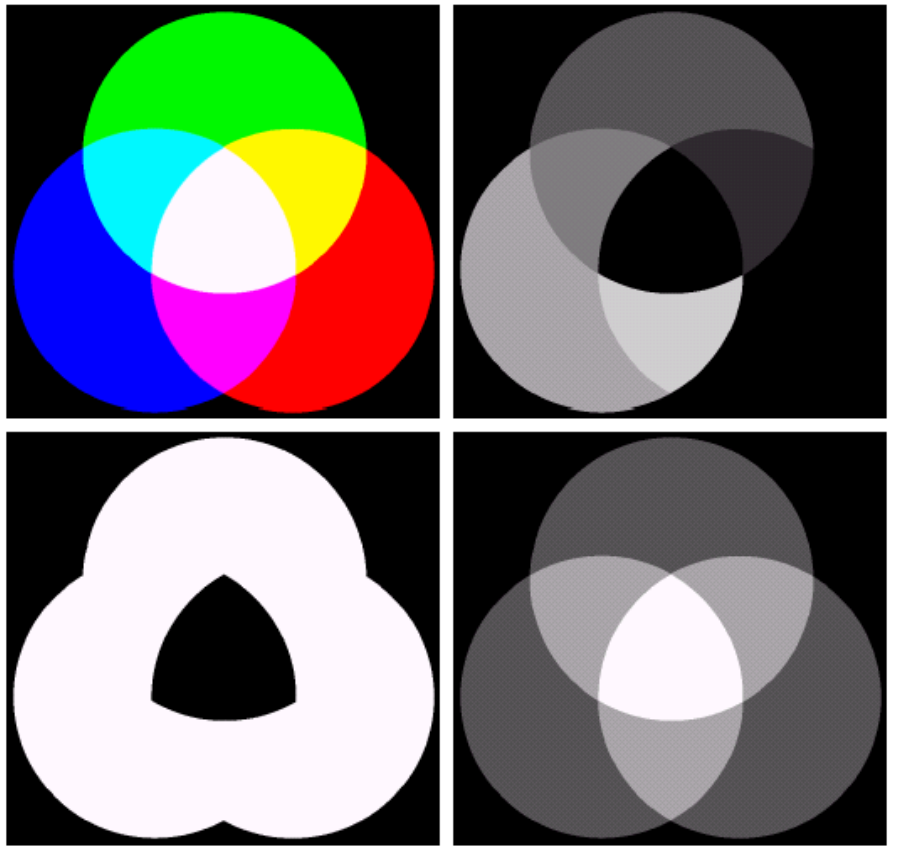
a b c

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

Converting colors from HSI to RGB

- Manipulating HSI component images

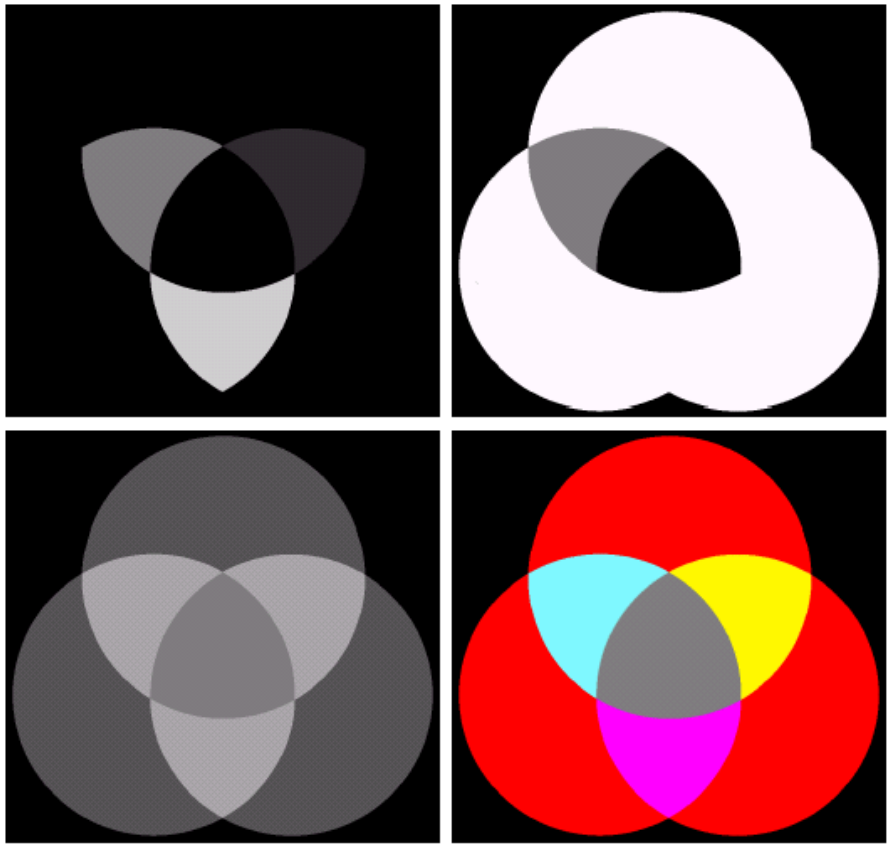




a b
c d

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

Converting colors from HSI to RGB

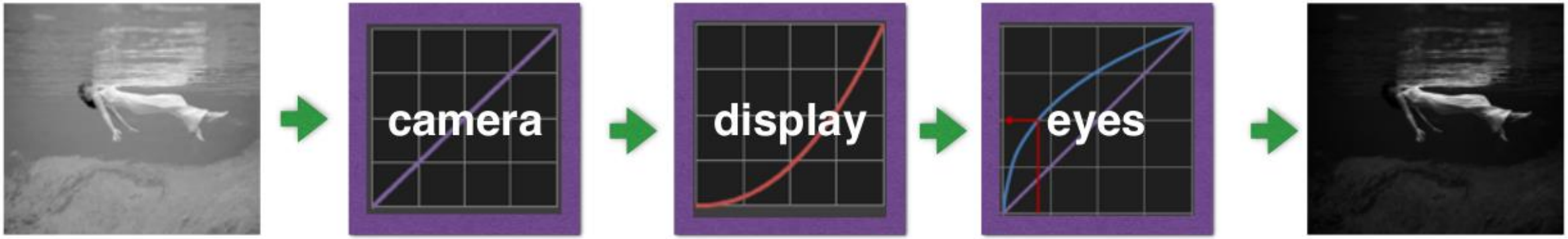


a b
c d

FIGURE 6.17 (a)–(c) Modified HSI component images. (d) Resulting RGB image. (See Fig. 6.16 for the original HSI images.)

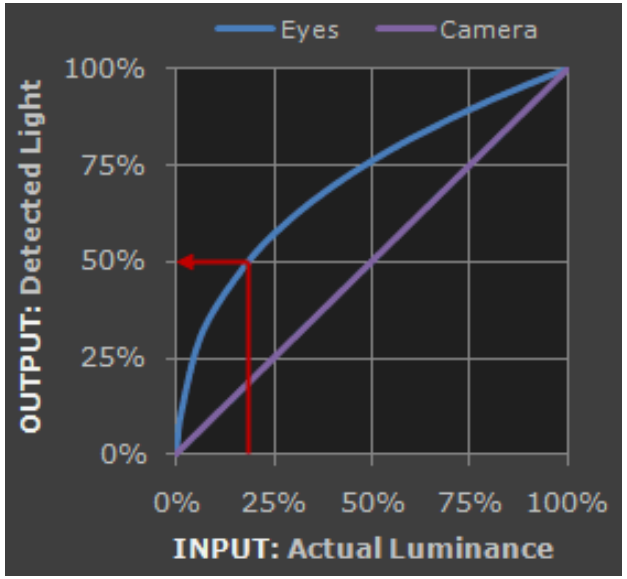
Tone reproduction

- Also known as gamma encoding (and erroneously as gamma correction).
- Without tone reproduction, images look very dark.



Why does this happen?

Perceived vs measured brightness by human eye



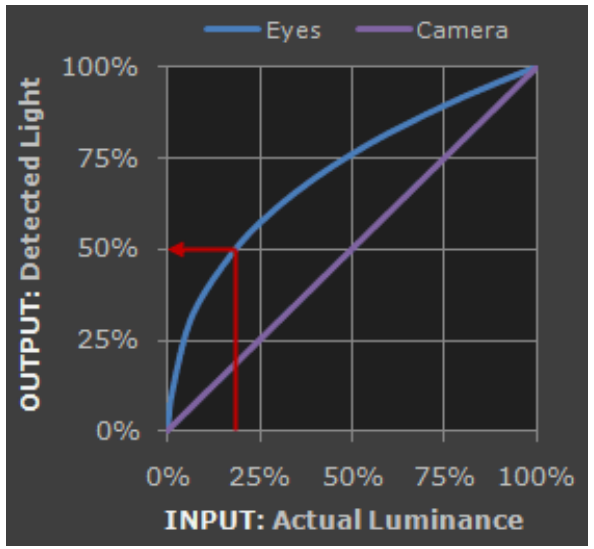
We have already seen that sensor response is linear.

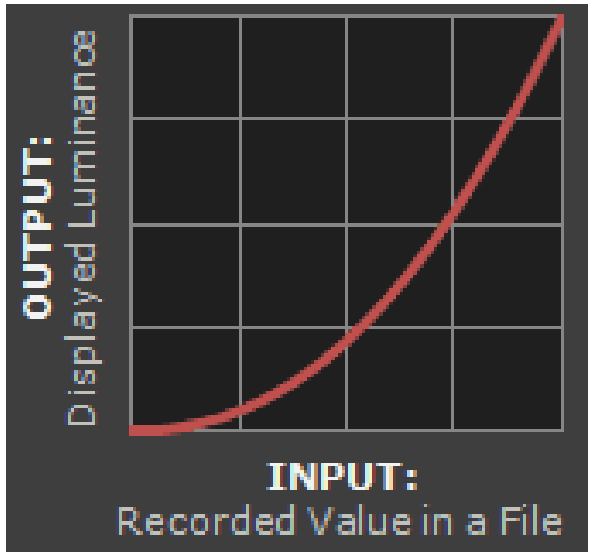
Human-eye *response* (measured brightness) is also linear.

However, human-eye *perception* (perceived brightness) is *non-linear*:

- More sensitive to dark tones.
- Approximately a Gamma function.

What about displays?





We have already seen that sensor response is linear.

Human-eye *response* (measured brightness) is also linear.

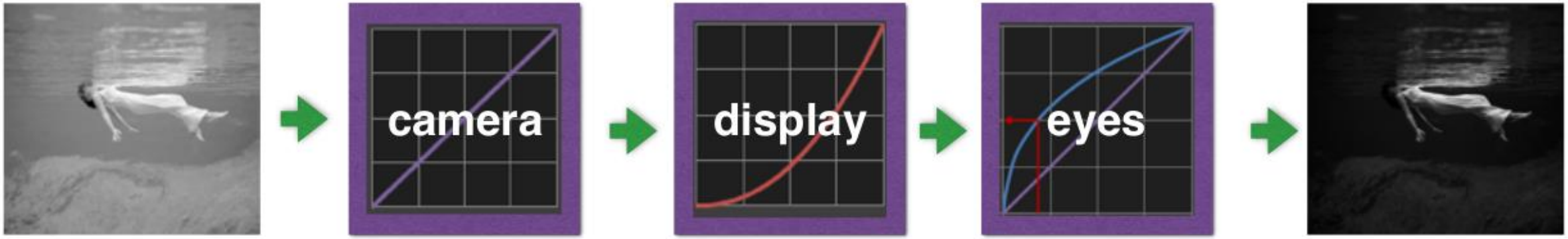
However, human-eye *perception* (perceived brightness) is *non-linear*:

- More sensitive to dark tones.
- Approximately a Gamma function.

Displays have a response opposite to that of human perception.

Tone reproduction

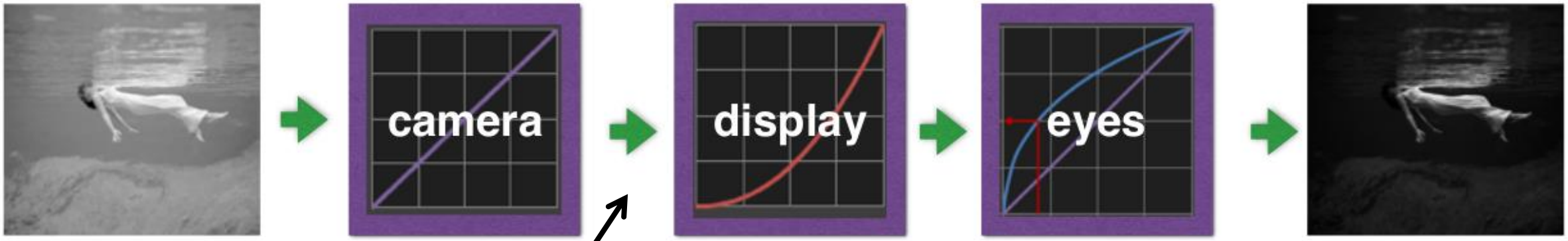
- Because of mismatch in displays and human eye perception, images look very dark.

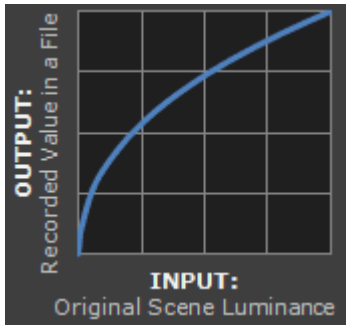


How do we fix this?

Tone reproduction

- Because of mismatch in displays and human eye perception, images look very dark.





- Pre-emptively cancel-out the display response curve.
- Add inverse display transform here.
- This transform is the tone reproduction or gamma correction.

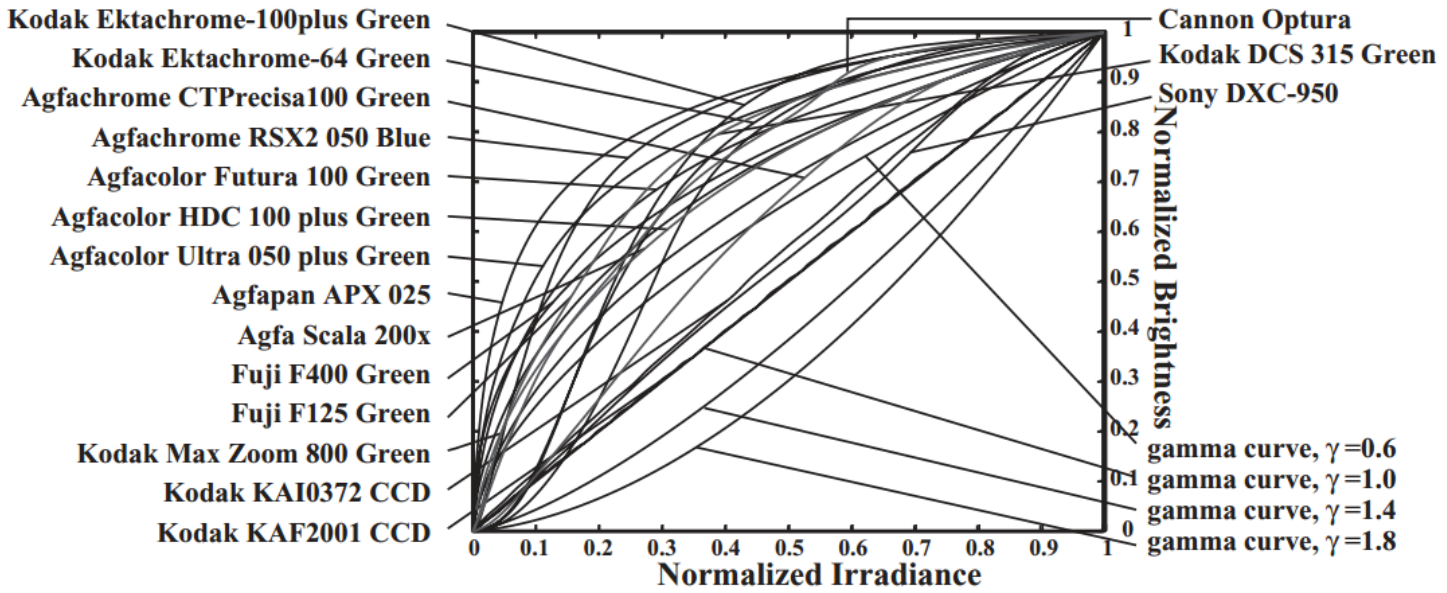
Tone reproduction curves

The exact tone reproduction curve depends on the camera.

- Often well approximated as L^{γ}, for different values of the power γ ("gamma").
- A good default is γ = 2.2.

before gamma

after gamma



Warning: Our values are no longer linear relative to scene radiance!

Histogram processing

- **Histogram**: is a discrete function that counts how many pixels have a given intensity value.

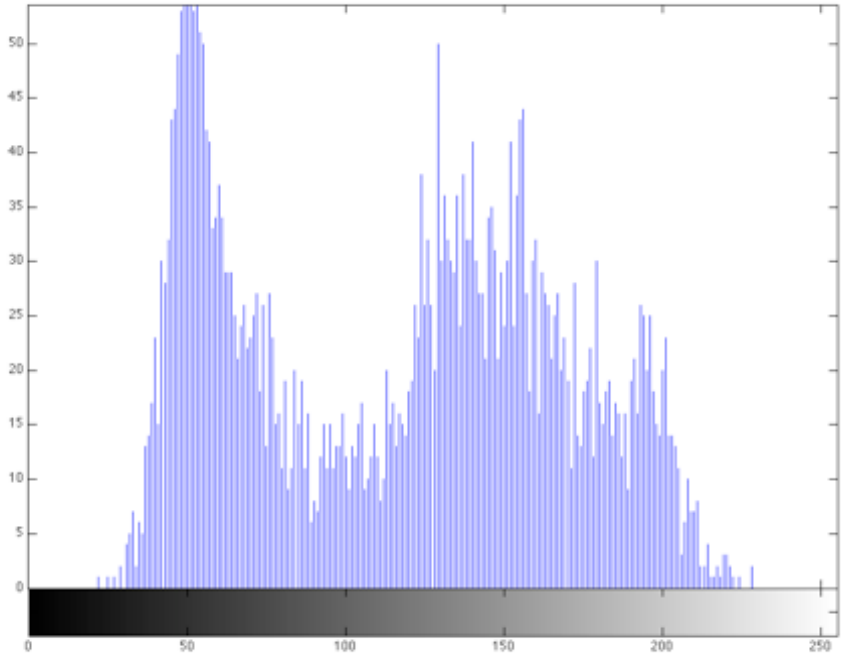
$$h(r_k) = n_k$$

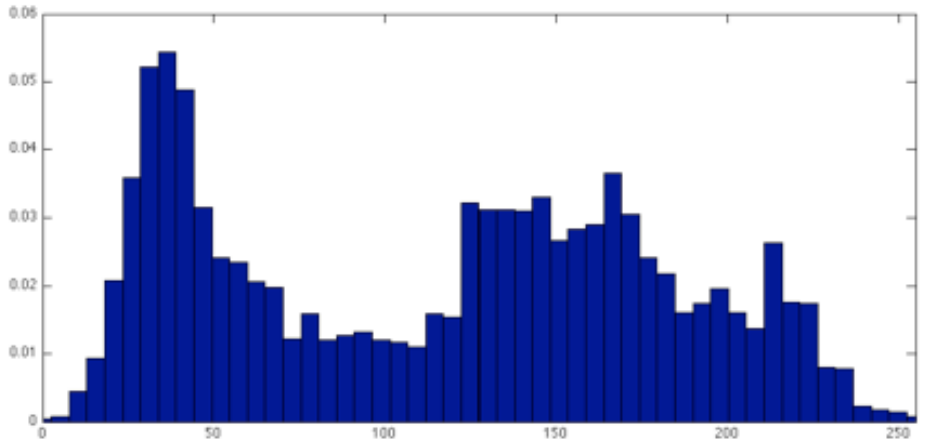
Number of pixels that have such intensity value

k-th intensity value

Histogram processing

- It is typical to scale n_k by *MN,* the number of pixels. The corresponding number represents the probability of a pixel having the given intensity.





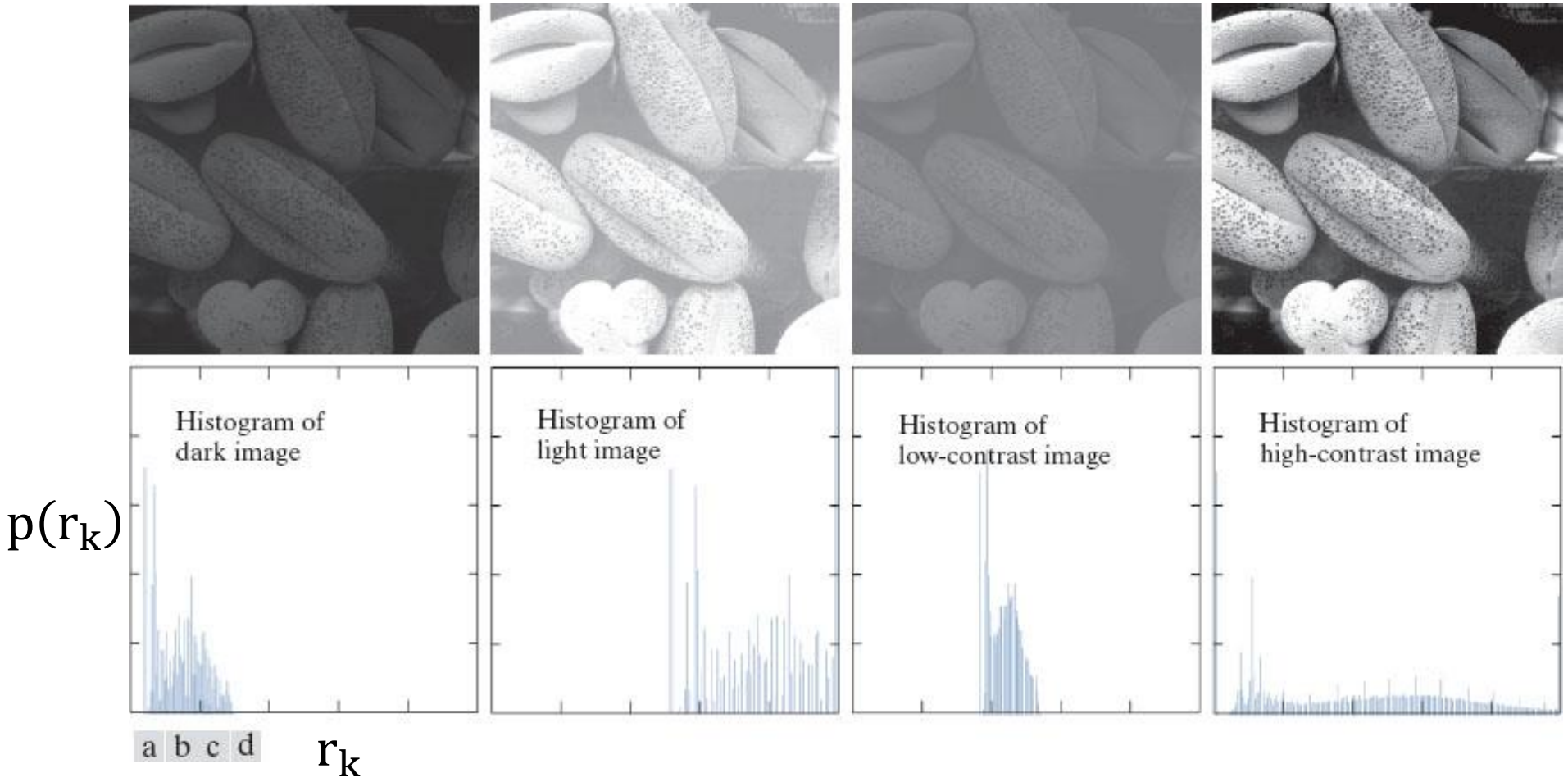
$$p(r_k) = \frac{n_k}{MN} \qquad \sum_k p(r_k) = 1$$

$$\sum_k n_k = MN$$

Probability that a random pixels has intensity r_k

Histogram processing

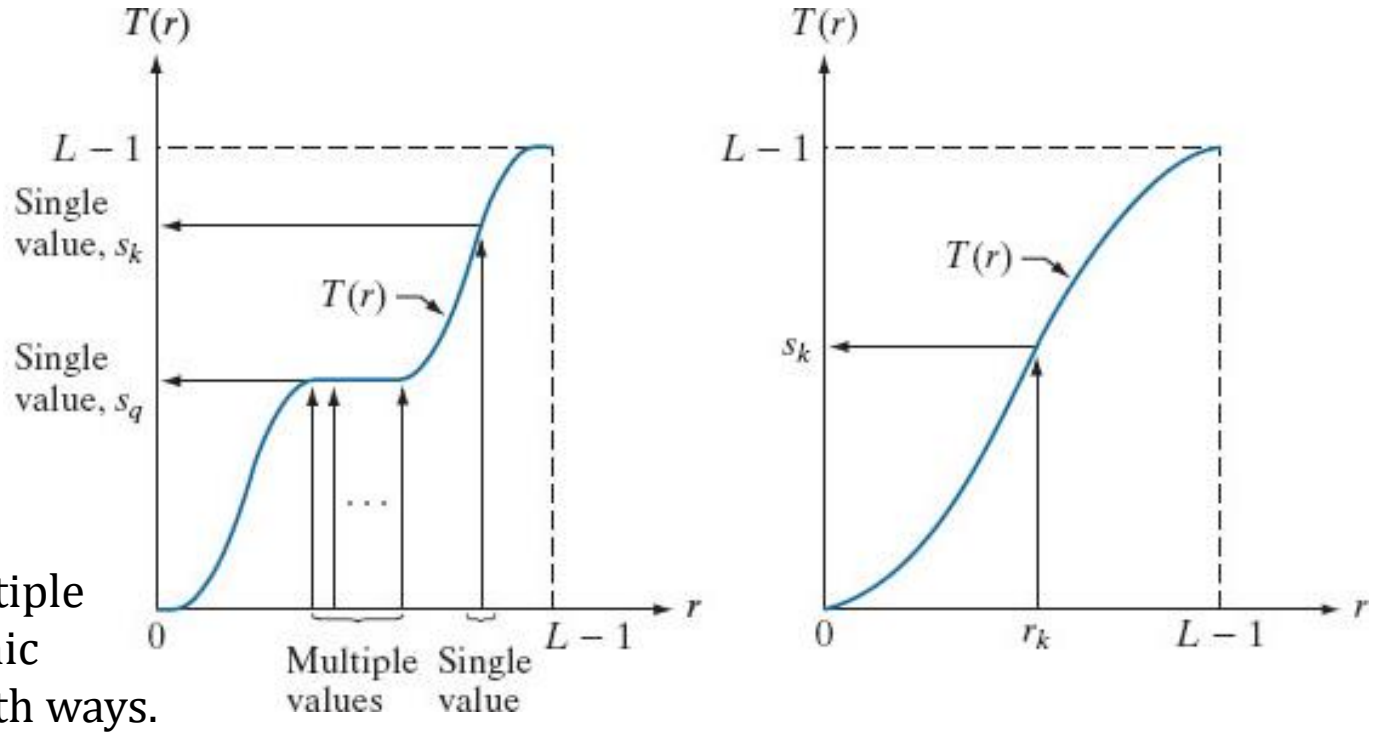
- (a) dark; (b) light; (c) low contrast; (d) high contrast.



Histogram Equalization

- Consider an intensity transformation $s = T(r)$ (r input intensity level, s output intensity level). Let us assume that
  - T is monotonically increasing that is $T(r_1) \leq T(r_2)$ for $r_1 \leq r_2$

    (strictly increasing if T^{-1} is required)
  - T maps [0, L − 1] to [0, L − 1]



(a) Monotonic increasing function, showing how multiple values can map to a single value. (b) Strictly monotonic increasing function. This is a one-to-one mapping, both ways.

Histogram Equalization

We can identify the intensity level to a *random variable* in $[0, L-1]$.
Then the function $p = h/MN$ is the probability density function of such random variable.

Denote by

- p_r the probability distribution of the original image (r-variable) and
- p_s the probability distribution of the image with intensity $s = T(r)$.

These two distributions will generally be different.

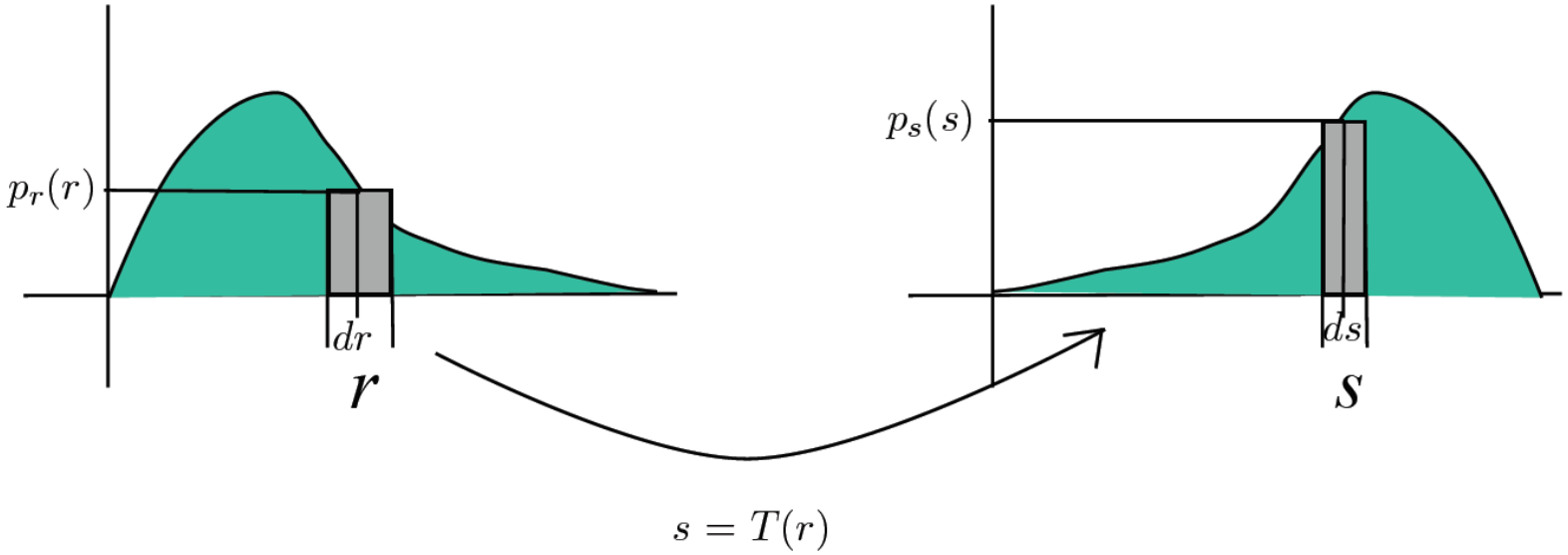
Thus, given the *change of variables* $s = T(r)$, the relation between p_s and p_r is given by the formula

$$p_s(s) = p_r(r) \left| \frac{dr}{ds} \right| .$$

Histogram Equalization

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right|.$$

Where does it come from?



(Recall, probability distributions are positive)

The gray regions must have the same area (we are just relabeling the variables)

$$p_r(r)dr = p_s(s)ds$$

(take the absolute value is for the case when T is decreasing)

Histogram Equalization

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right| \quad \Longleftrightarrow \quad p_s(s) = p_r(r)\frac{1}{|T'(r)|}$$

(Alternative formulation)

Example. Consider the transformation

$$s = T(r) = (L-1)\int_0^r p_r(w)dw.$$

The transformation is strictly increasing because p_r is a positive function.

Scaling factor so that *T(L-1) = L-1*

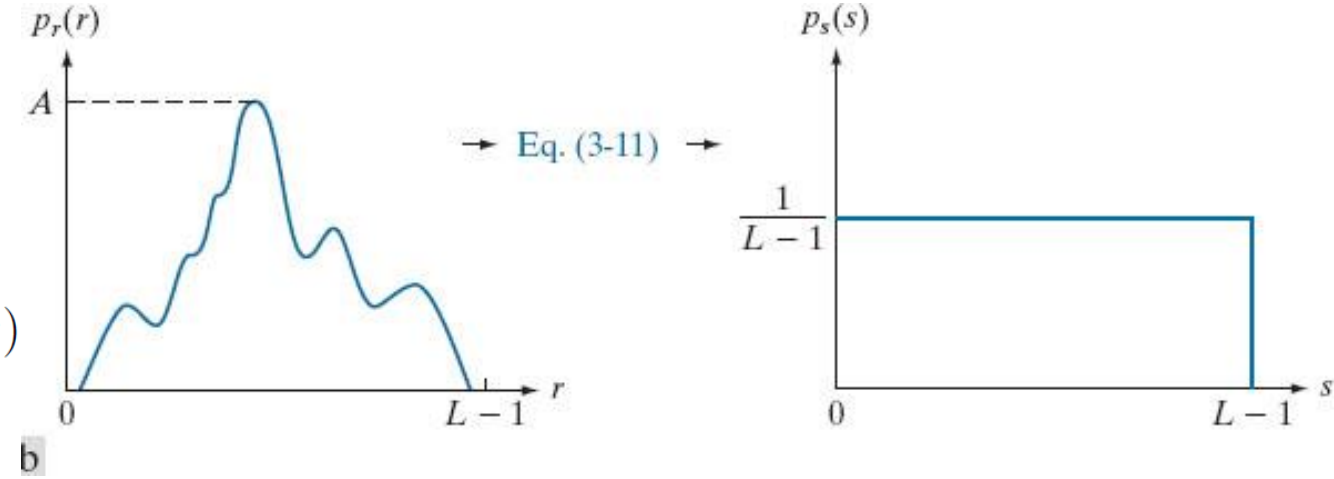
Cumulative probability

Histogram Equalization

What is $p_s(s)$?

Let's use the formula $p_s(s) = p_r(r)\frac{1}{|T'(r)|}$

(a) An arbitrary PDF. (b) Result of applying Eq. (3-11) to the input PDF. The resulting PDF is always uniform, independently of the shape of the input.



We derive $T(r)$ with respect to r

$$T'(r) = \frac{d}{dr}T(r) = (L-1)p_r(r)$$

Hence

$$p_s(s) = p_r(r)\frac{1}{|T'(r)|} = p_r(r)\frac{1}{(L-1)p_r(r)} = \frac{1}{L-1}$$

The uniform distribution!

Histogram Equalization

For discrete density functions, the equivalent of

$$s = T(r) = (L-1)\int_0^r p_r(w)dw.$$

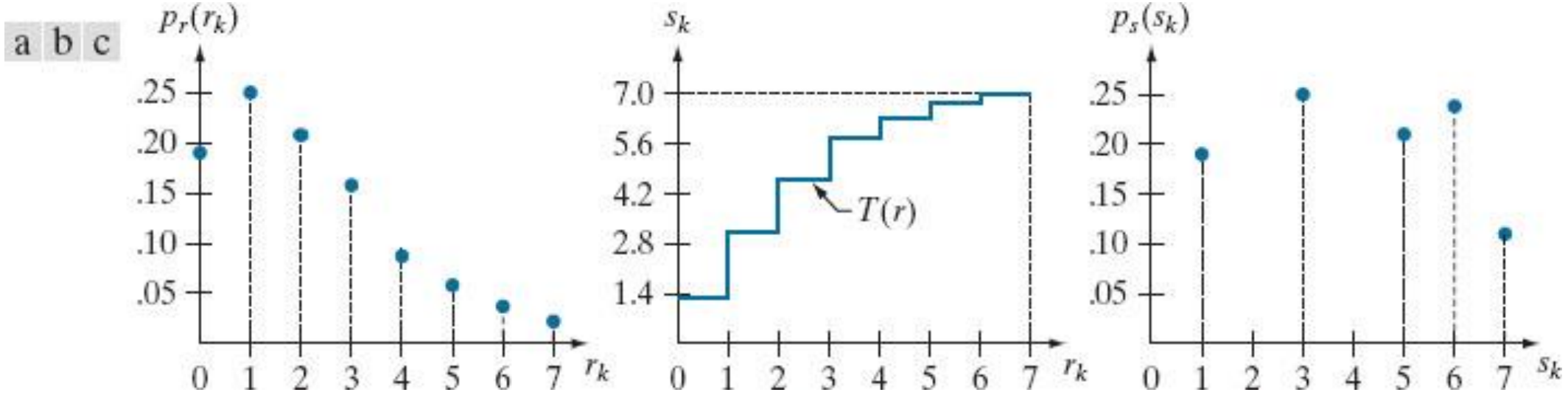
is obtained by replacing the integral by a sum,

$$\begin{aligned} s_k &= T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) \\ &= \frac{L-1}{MN}\sum_{j=0}^{k} n_j \qquad k = 0, 1, \ldots, L-1. \end{aligned}$$

This transformation is called *histogram equalization/linearization.*

Histogram Equalization

- Since the s_k is an intensity transformation, it must be an integer value. This rounding causes the resulting histogram not necessarily to result into a uniform histogram

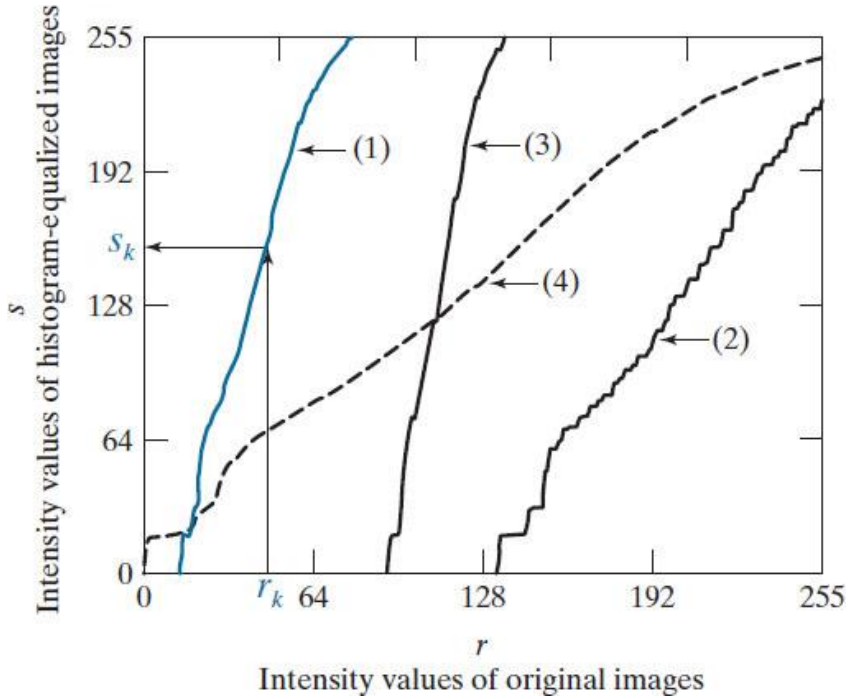


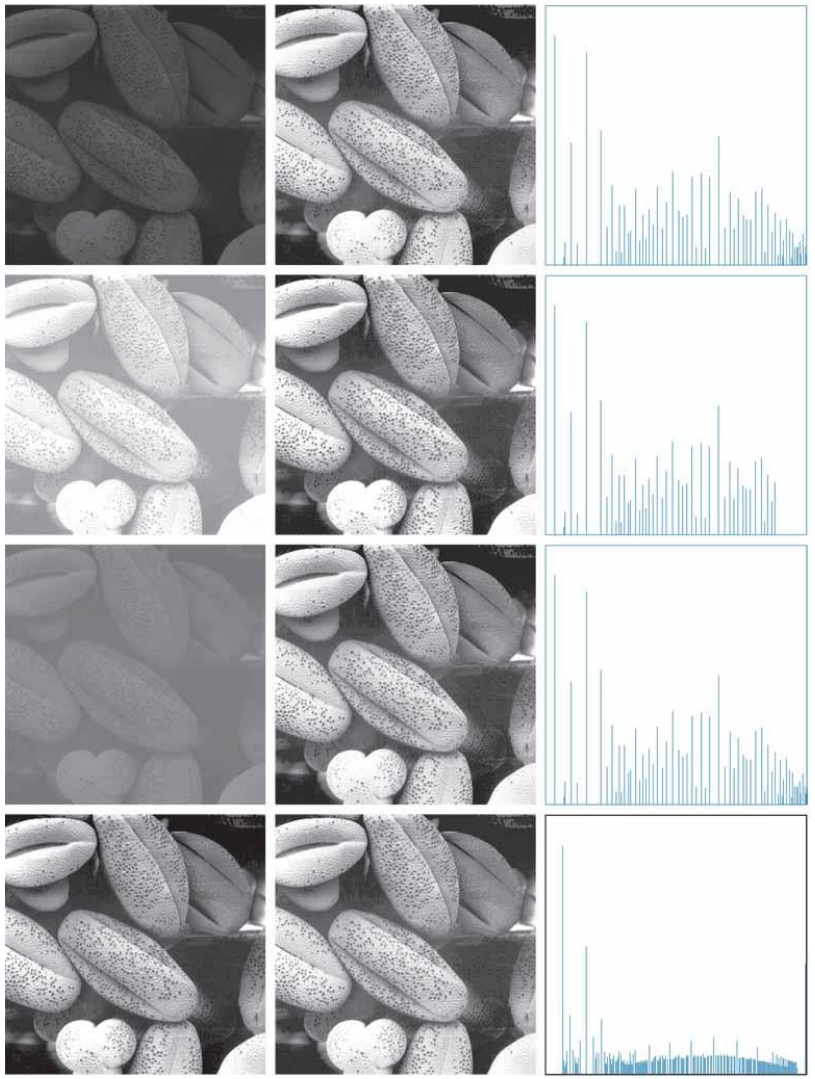
(a) Original histogram. (b) Transformation function. (c) Equalized histogram

- Though the resulting histogram is not uniform, the final result is that the intensity values are spread out (span a wider range)

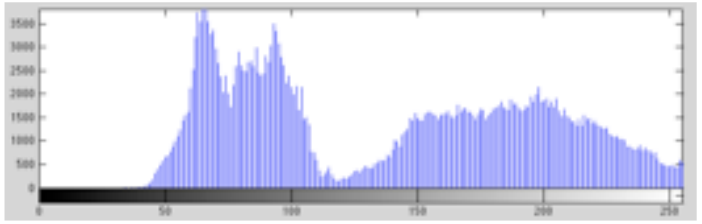
Histogram equalization results

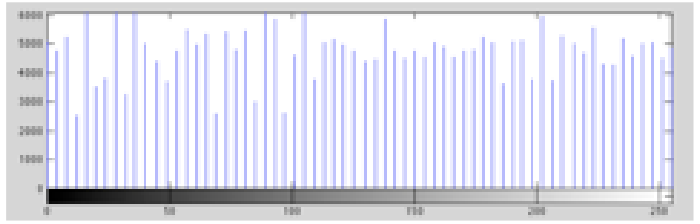
- Left column: Input images. Center column: Corresponding histogram-equalized images. Right column: histograms of the images in the center column



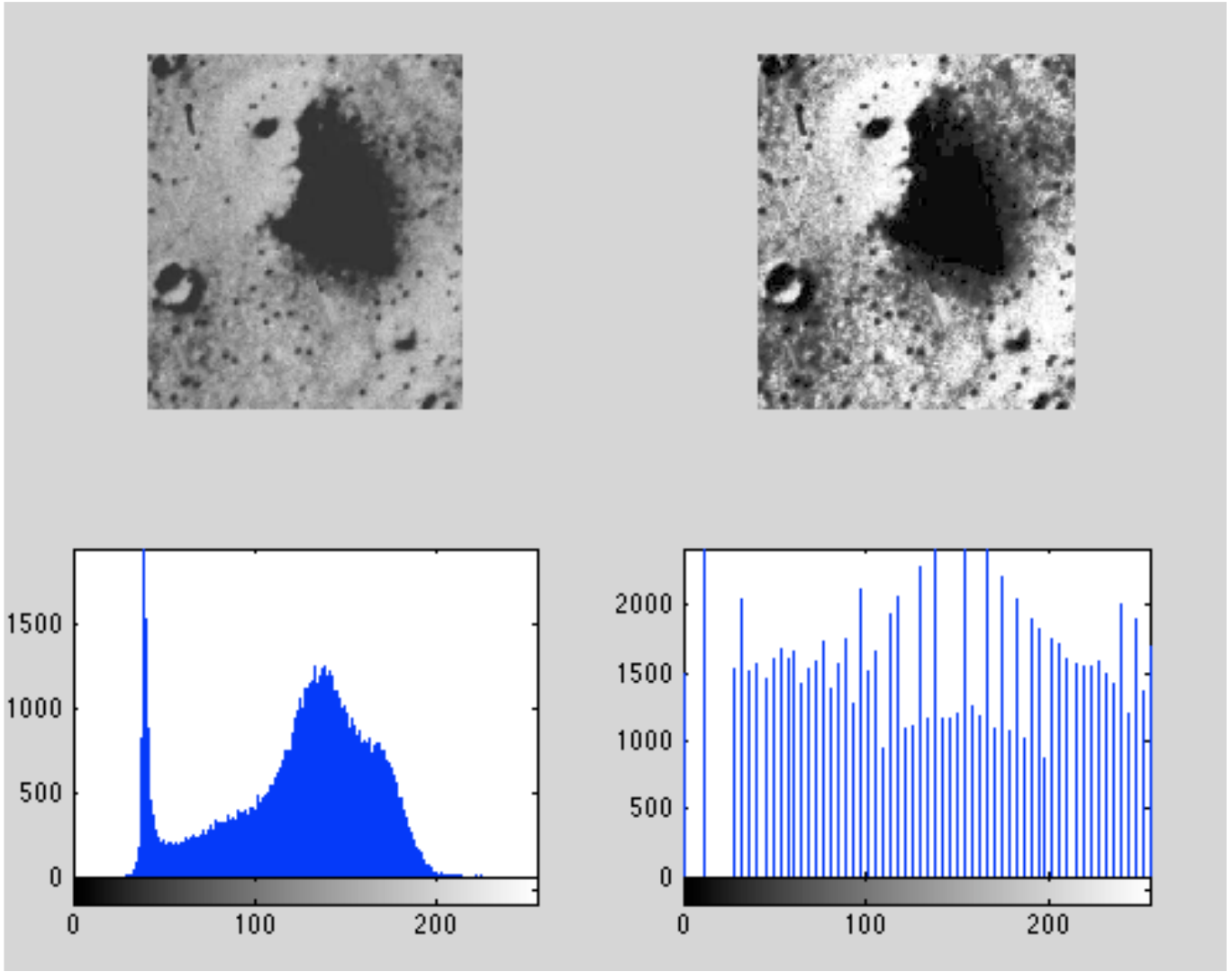


Histogram equalization results

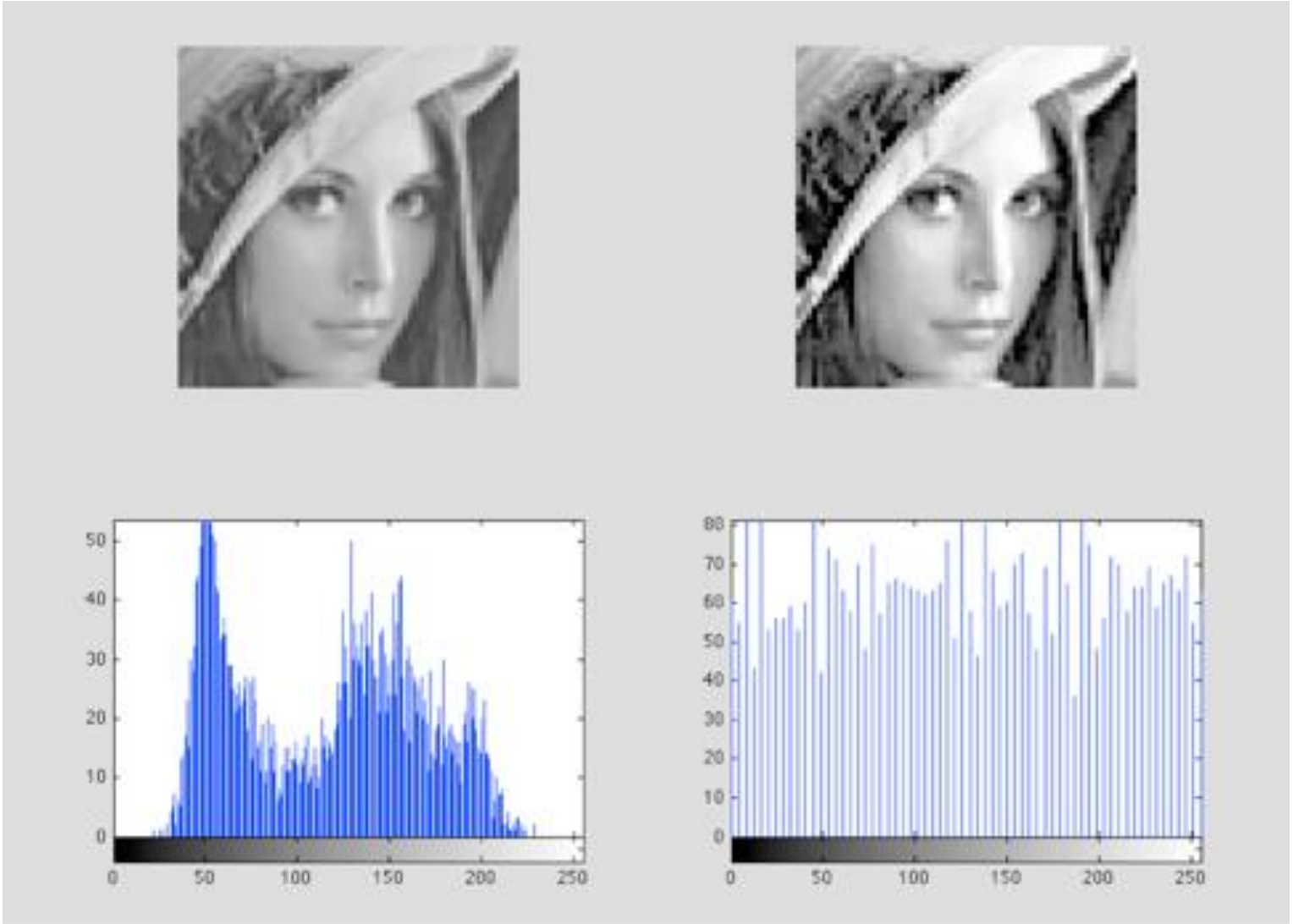




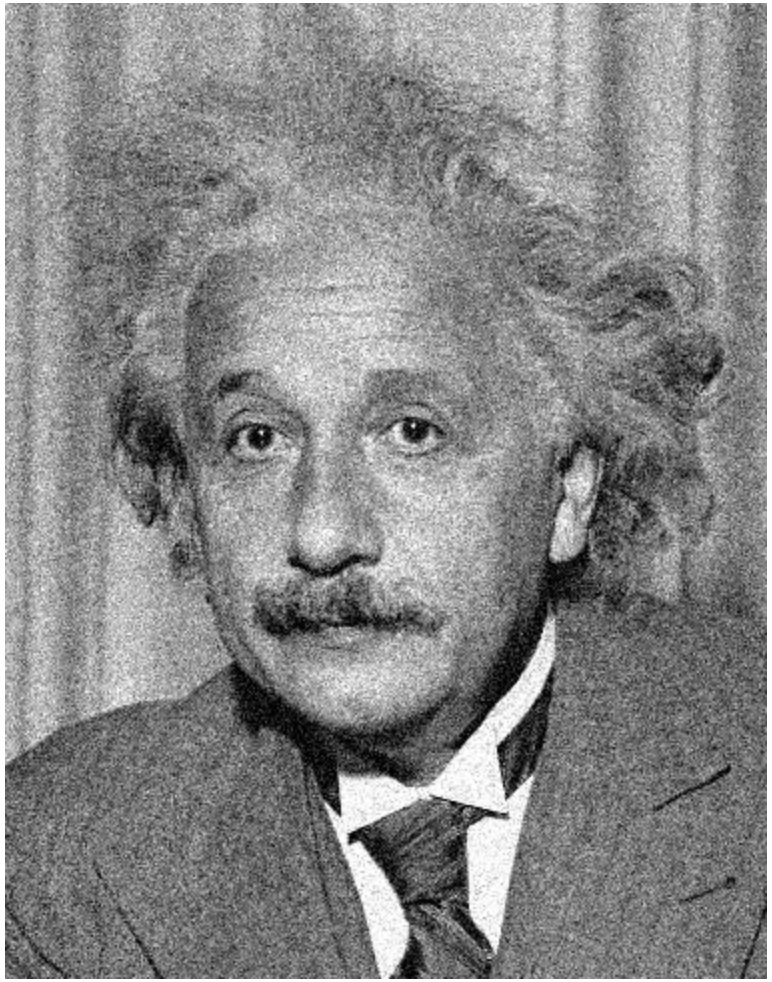
Histogram equalization results



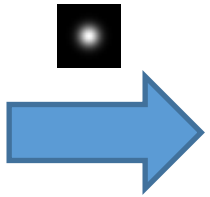
Histogram equalization results



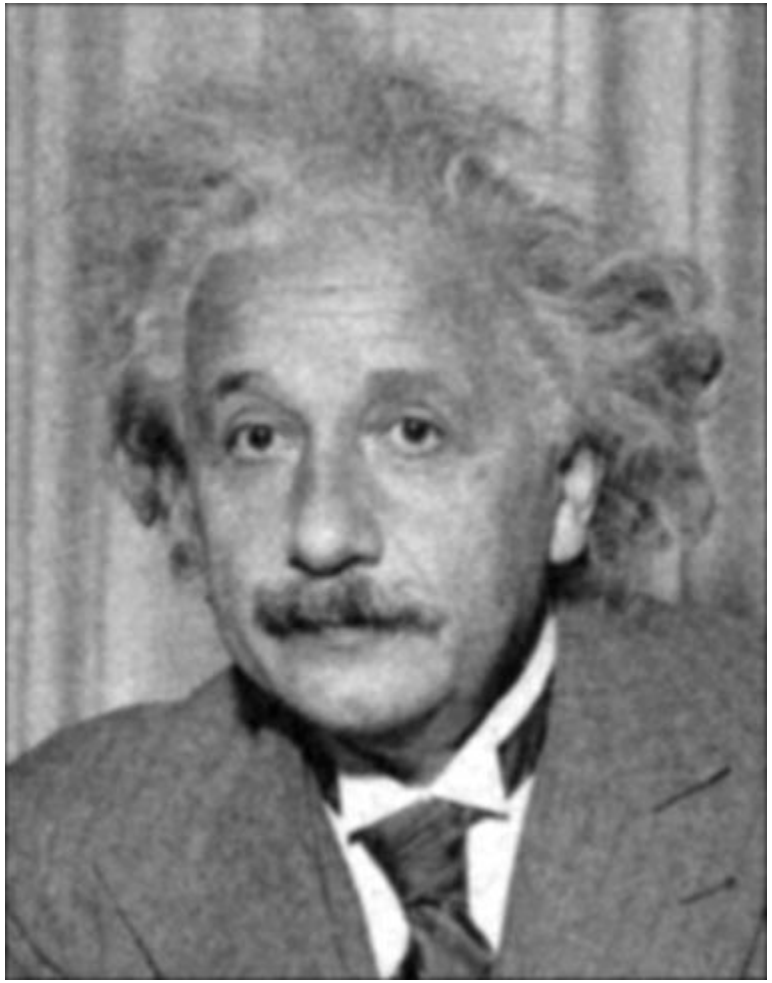
Denoising



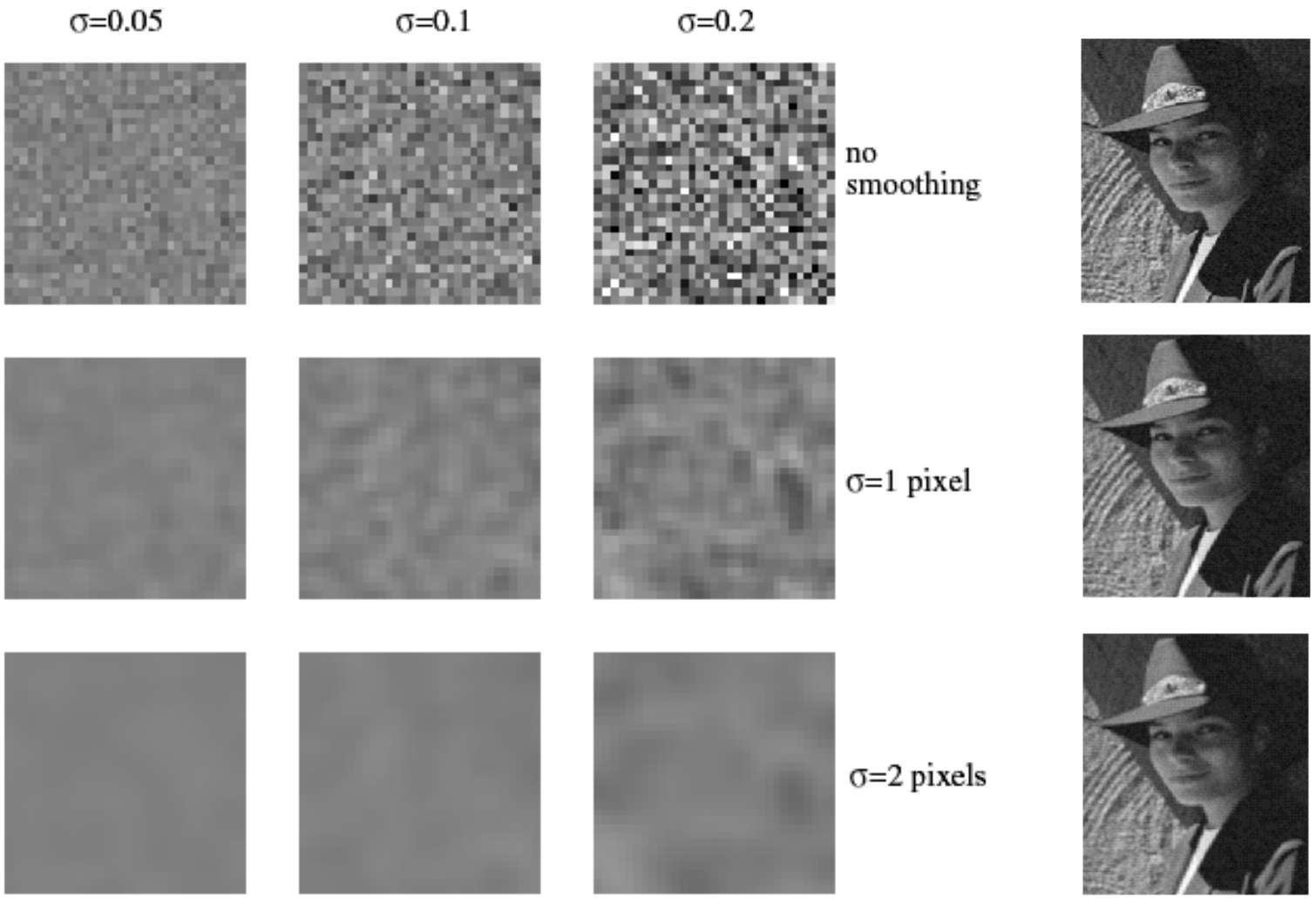
Additive Gaussian Noise



Gaussian Filter



Reducing Gaussian noise



Smoothing with larger standard deviations suppresses noise, but also blurs the image

Source: S. Lazebnik

Reducing salt-and-pepper noise by Gaussian smoothing

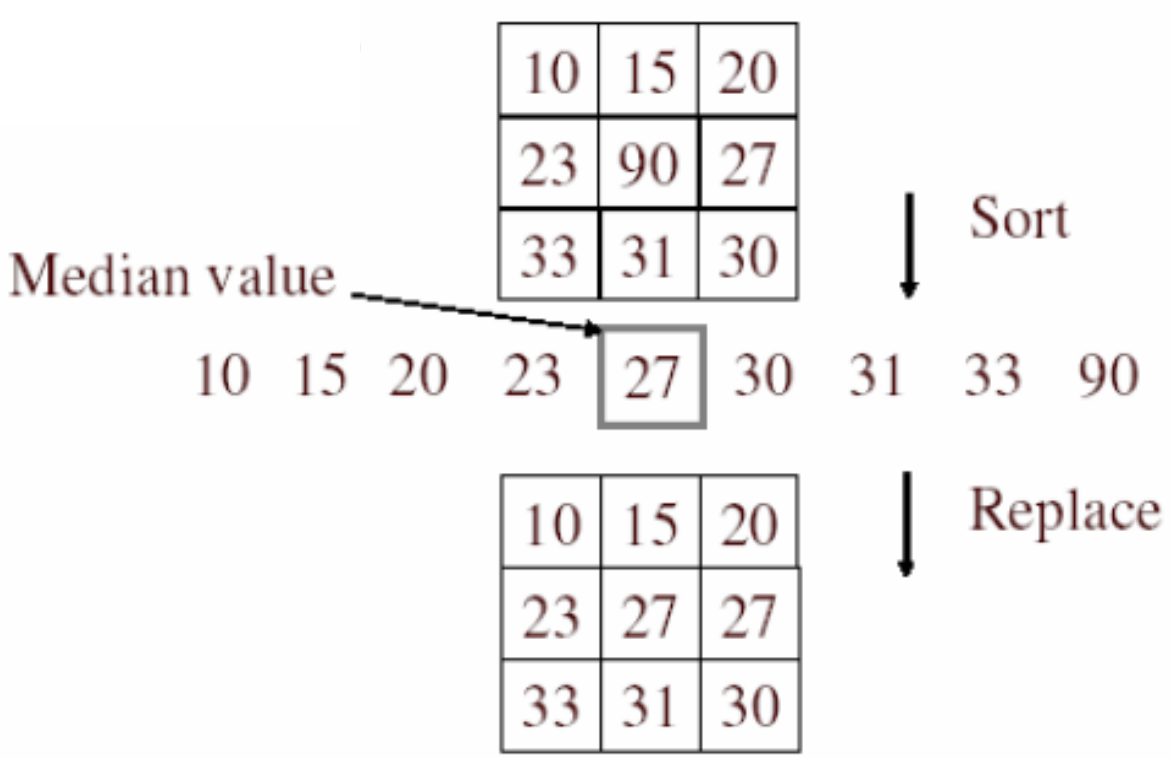
3x3

5x5

7x7

Alternative idea: Median filtering

- A **median filter** operates over a window by selecting the median intensity in the window

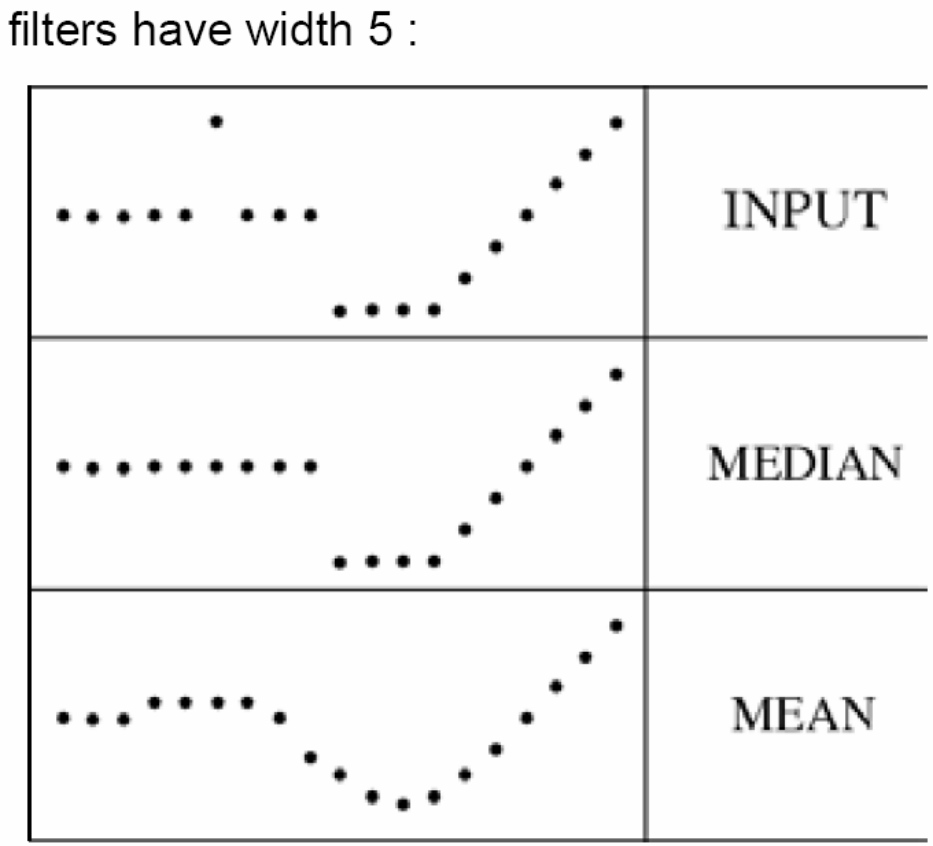


- Is median filtering linear?

Source: K. Grauman

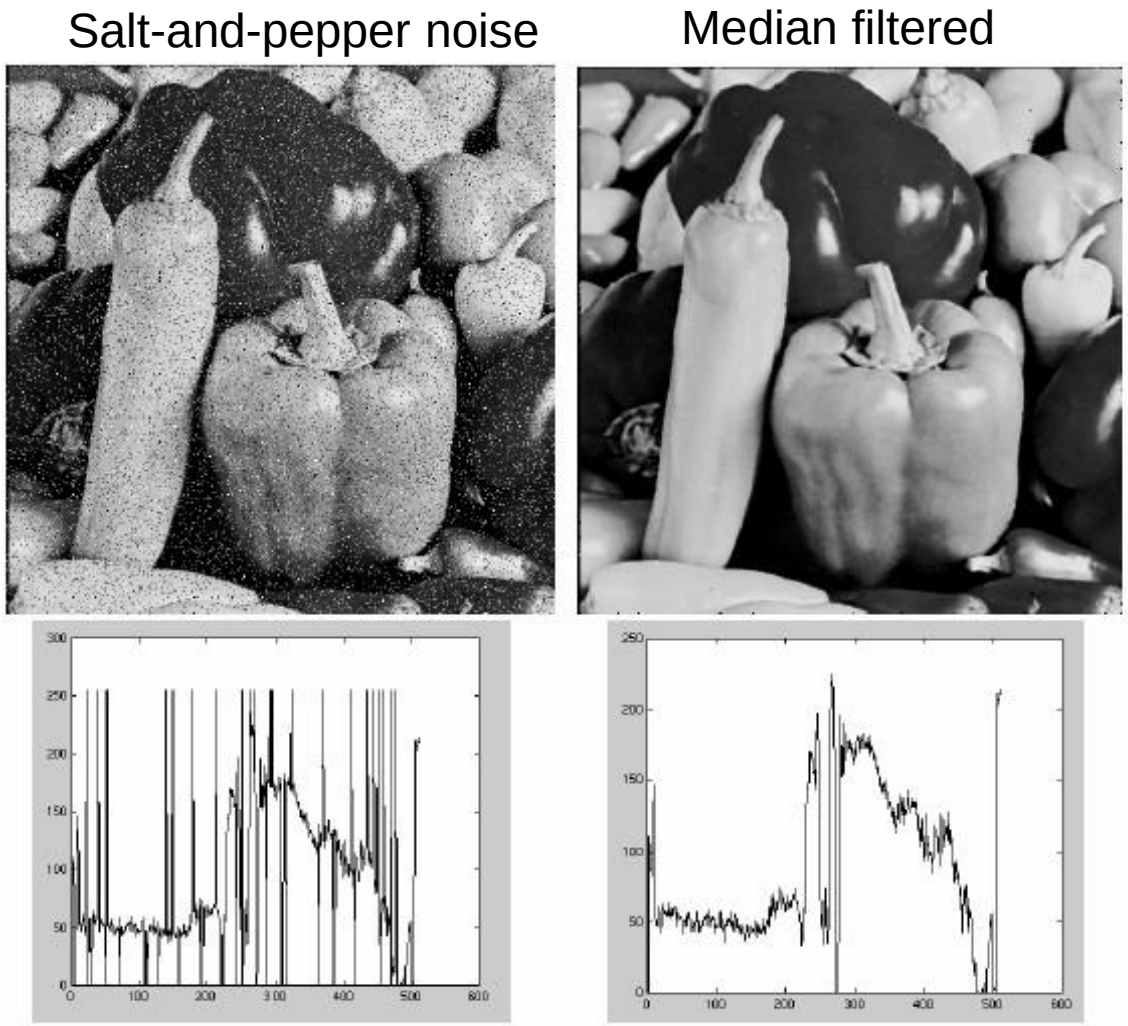
Median filter

- What advantage does median filtering have over Gaussian filtering?
  - Robustness to outliers



Source: K. Grauman

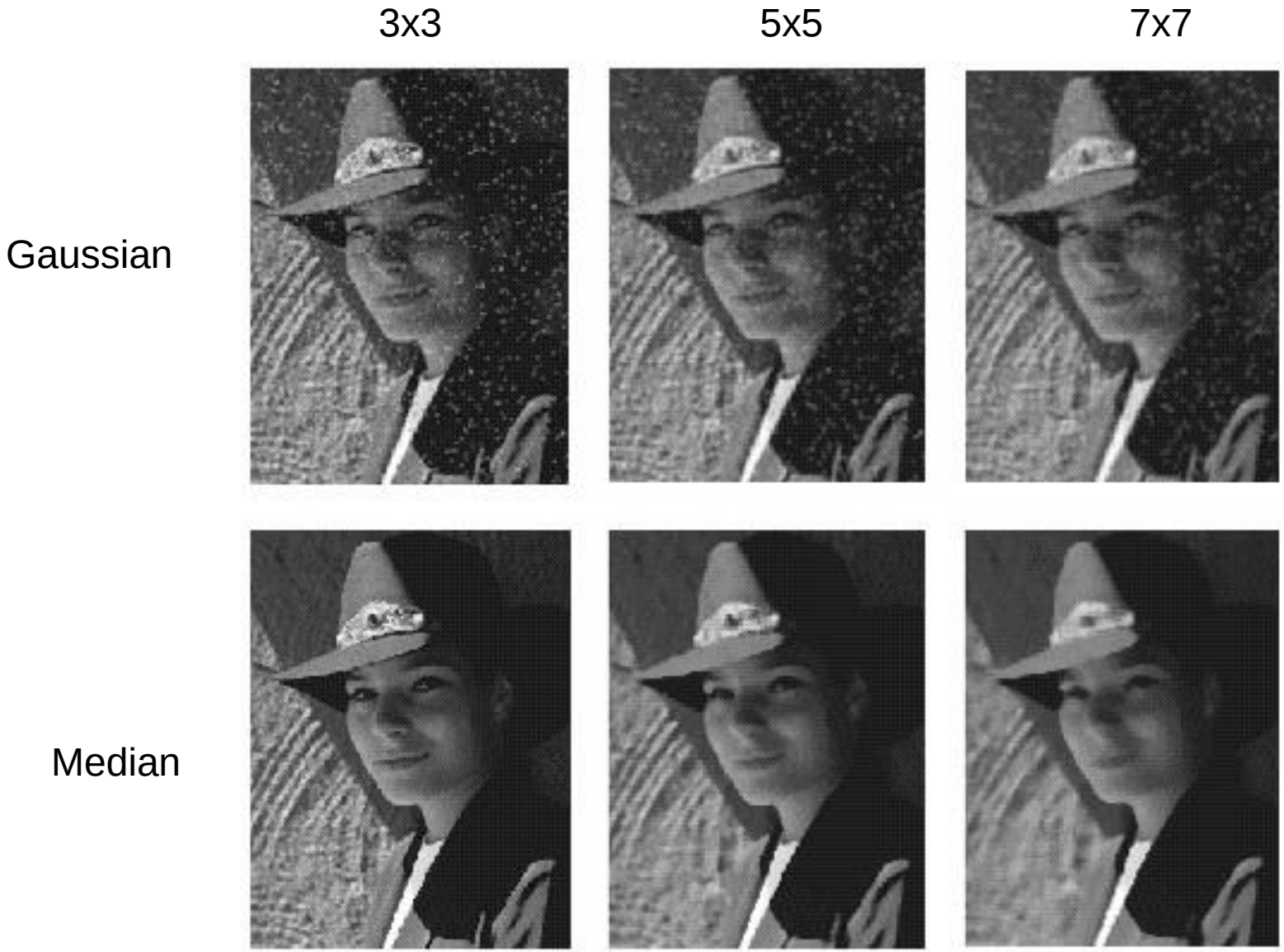
Median filter



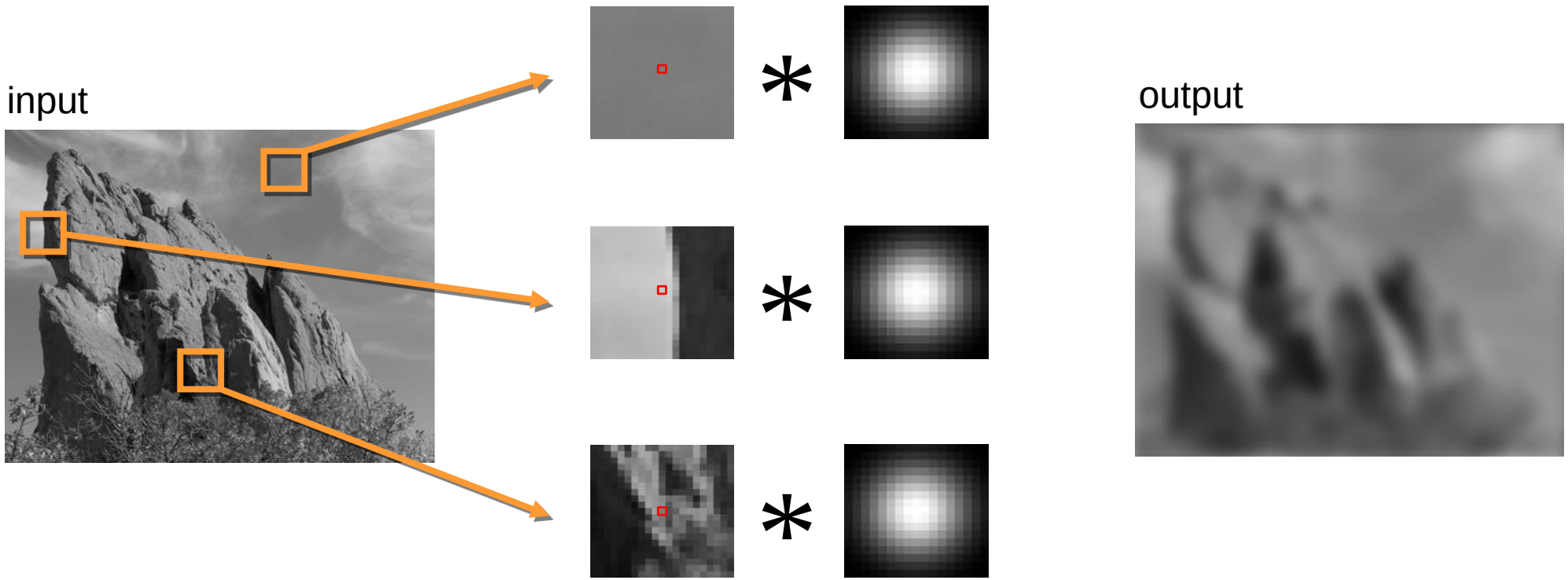
- MATLAB: medfilt2(image, [h w])

Source: M. Hebert

Median vs. Gaussian filtering

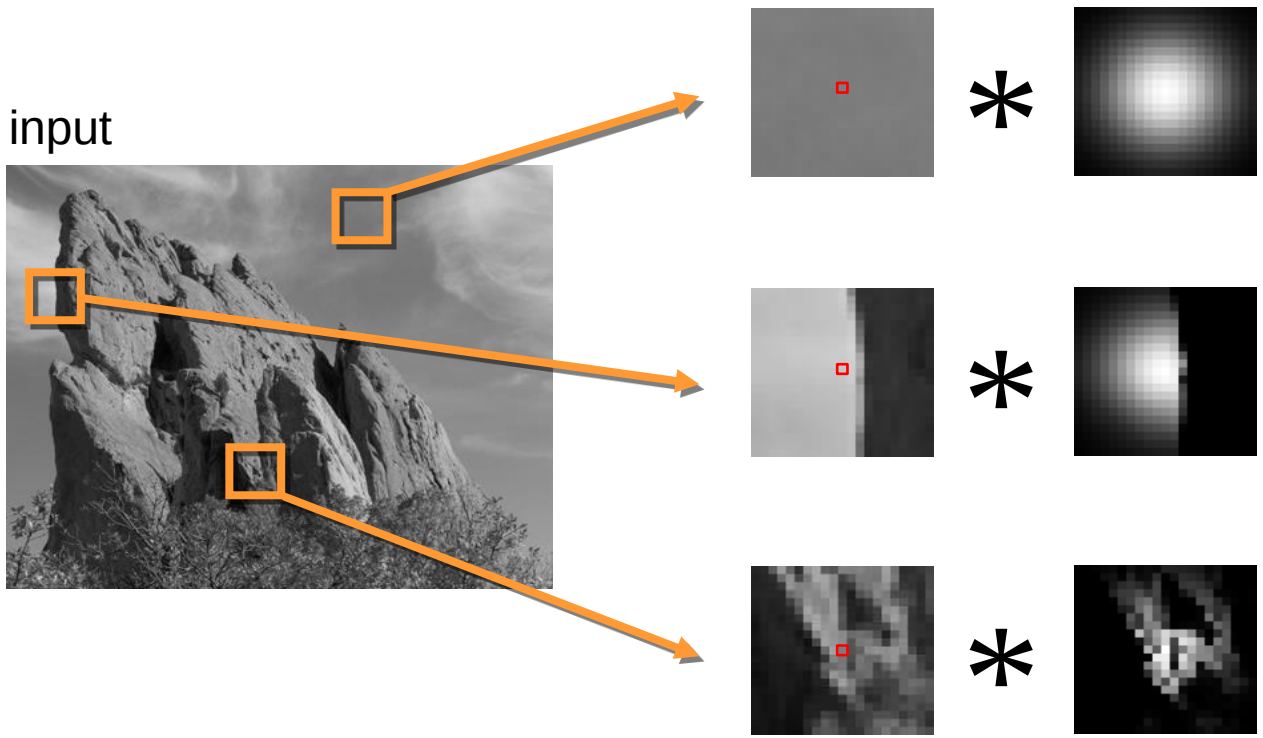


Gaussian smoothing



Same Gaussian kernel everywhere
Averages across edges ⇒ blur

Bilateral filtering



Kernel shape depends on image content
Avoids averaging across edges

Gaussian smoothing

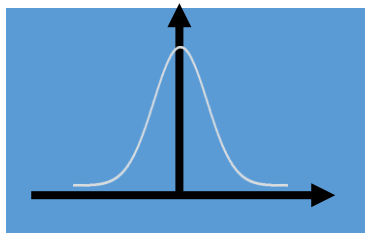
- Does smooth images
- But smoothes too much: **edges are blurred**.
  - Only spatial distance matters
  - No edge term

$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q}\in S} G_{\sigma}(||\mathbf{p}-\mathbf{q}||)\, I_{\mathbf{q}}$$

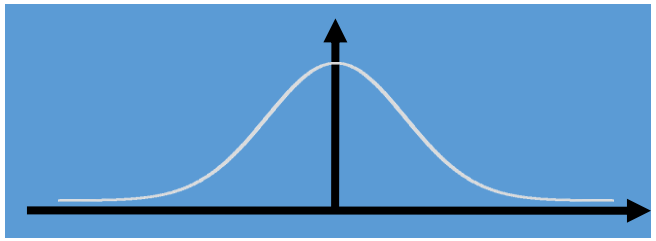
space

$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q}\in S} G_{\sigma}(||\mathbf{p}-\mathbf{q}||)\, I_{\mathbf{q}}$$

size of the window



small σ



large σ

limited smoothing

strong smoothing

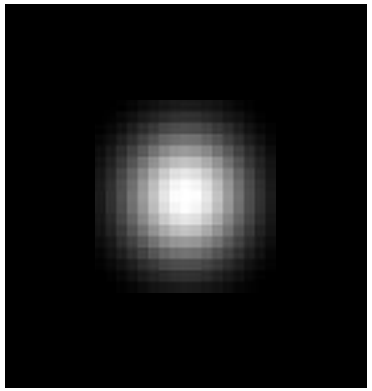
Bilateral filtering

- Same idea: **weighted average of pixels**.

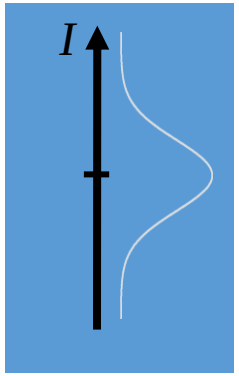
$$BF[I]_{\mathbf{p}} = \underbrace{\frac{1}{W_{\mathbf{p}}}}_{\text{new}} \sum_{\mathbf{q}\in S} \underbrace{G_{\sigma_s}(||\mathbf{p}-\mathbf{q}||)}_{\text{not new}} \underbrace{G_{\sigma_r}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)}_{\text{new}} I_{\mathbf{q}}$$

normalization factor

space weight

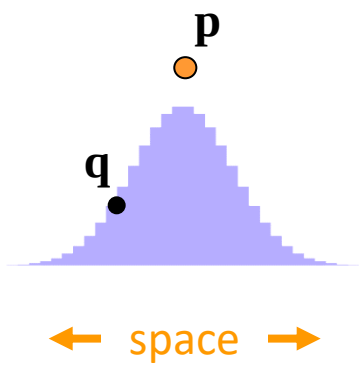


range weight



Gaussian Blur and Bilateral Filter

Gaussian blur

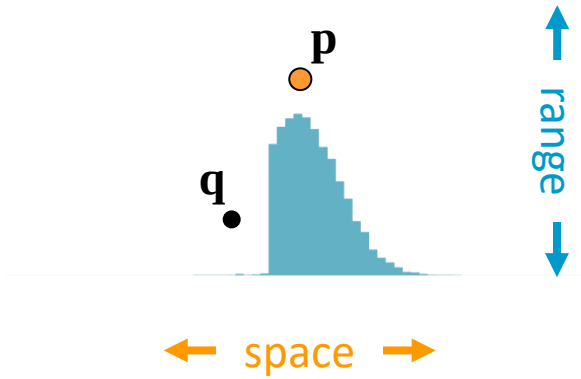


$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q}\in S} G_{\sigma}(||\mathbf{p}-\mathbf{q}||)\, I_{\mathbf{q}}$$

space

Bilateral filter

[Aurich 95, Smith 97, Tomasi 98]



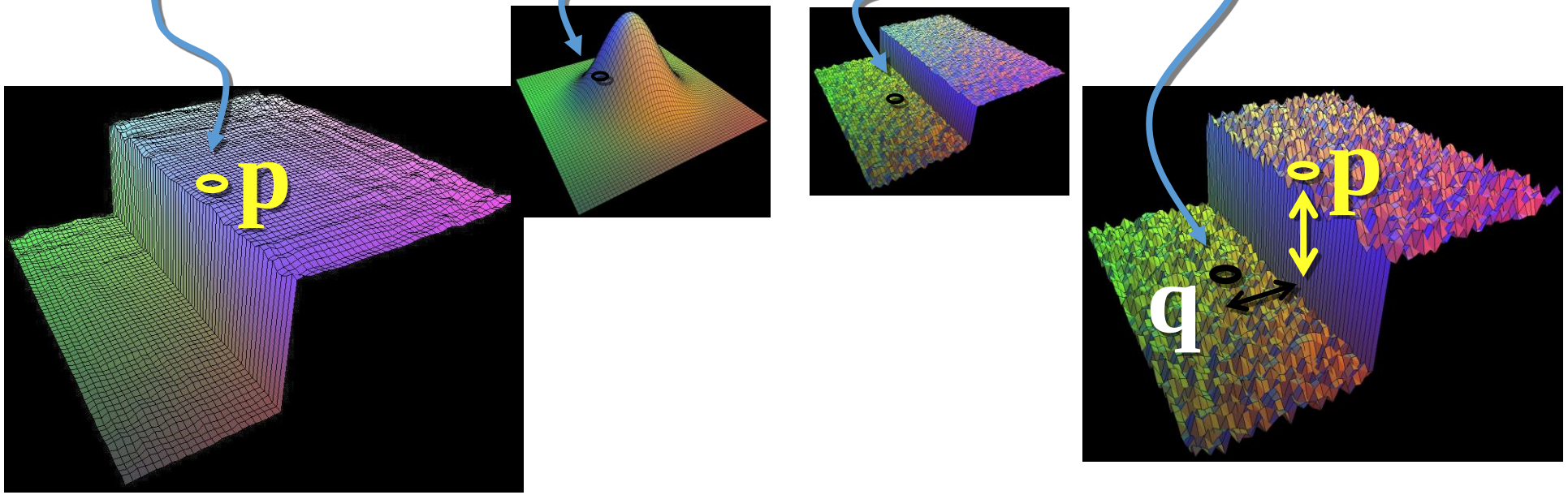
$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}\in S} G_{\sigma_s}(||\mathbf{p}-\mathbf{q}||)\, G_{\sigma_r}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)\, I_{\mathbf{q}}$$

space range

normalization

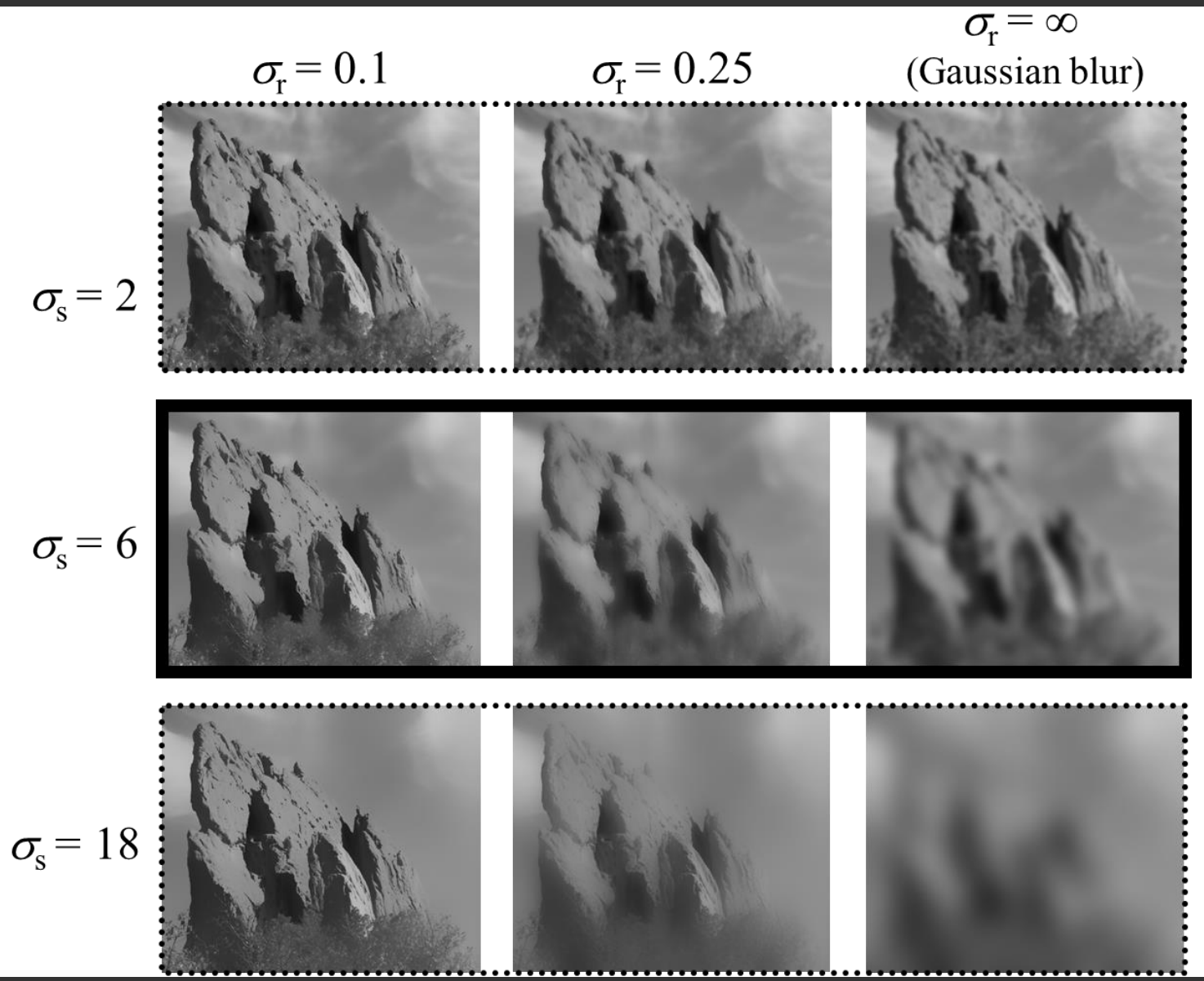
Bilateral filter on a height field

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|) \; G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|) \; I_{\mathbf{q}}$$



Varying parameters

input



Matrix manipulating

```python
import cv2, numpy as np

image = np.full((480, 640, 3), 255, np.uint8)
cv2.imshow('white', image)
cv2.waitKey()
cv2.destroyAllWindows()

image = np.full((480, 640, 3), (0, 0, 255), np.uint8)
cv2.imshow('red', image)
cv2.waitKey()
cv2.destroyAllWindows()

image.fill(0)
cv2.imshow('black', image)
cv2.waitKey()
cv2.destroyAllWindows()
```

```python
image[240, 160] = image[240, 320] = image[240, 480] = (255, 255, 255)
cv2.imshow('black with white pixels', image)
cv2.waitKey()
cv2.destroyAllWindows()

image[:, :, 0] = 255
cv2.imshow('blue with white pixels', image)
cv2.waitKey()
cv2.destroyAllWindows()

image[:, 320, :] = 255
cv2.imshow('blue with white line', image)
cv2.waitKey()
cv2.destroyAllWindows()

image[100:600, 100:200, 2] = 255
cv2.imshow('image', image)
cv2.waitKey()
cv2.destroyAllWindows()
```

Converting data types

```python
import cv2
import numpy as np

image = cv2.imread('../data/Lena.png')
print('Shape:', image.shape)
print('Data type:', image.dtype)

cv2.imshow('image', image)
cv2.waitKey()
cv2.destroyAllWindows()

image = image.astype(np.float32) / 255
print('Shape:', image.shape)
print('Data type:', image.dtype)

cv2.imshow('image', np.clip(image*2, 0, 1))
cv2.waitKey()
cv2.destroyAllWindows()

image = (image * 255).astype(np.uint8)
print('Shape:', image.shape)
print('Data type:', image.dtype)

cv2.imshow('image', image)
cv2.waitKey()
cv2.destroyAllWindows()
```

Manipulating image channels

```
import cv2, numpy as np
image = cv2.imread('../data/Lena.png').astype(np.float32) / 255
print('Shape:', image.shape)
cv2.imshow('original image', image)

image[:, :, [0, 2]] = image[:, :, [2, 0]]
cv2.imshow('blue_and_red_swapped', image)
cv2.waitKey()

image[:, :, [0, 2]] = image[:, :, [2, 0]]
image[:, :, 0] = (image[:, :, 0] * 0.9).clip(0, 1)
image[:, :, 1] = (image[:, :, 1] * 1.1).clip(0, 1)
cv2.imshow('converted image', image)
cv2.waitKey()
cv2.destroyAllWindows()
```

Converting color space

```python
import cv2
import numpy as np

image = cv2.imread('../data/Lena.png').astype(np.float32) / 255
print('Shape:', image.shape)
print('Data type:', image.dtype)
cv2.imshow("original image", image)

gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
print('Converted to grayscale')
print('Shape:', gray.shape)
print('Data type:', gray.dtype)
cv2.imshow("gray-scale image", gray)

hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
print('Converted to HSV')
print('Shape:', hsv.shape)
print('Data type:', hsv.dtype)

hsv[:, :, 2] *= 2
from_hsv = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)
print('Converted back to BGR from HSV')
print('Shape:', from_hsv.shape)
print('Data type:', from_hsv.dtype)
cv2.imshow('from_hsv', from_hsv)
cv2.waitKey()
cv2.destroyAllWindows()
```

Gamma correction

```python
import cv2
import numpy as np

image = cv2.imread('../data/Lena.png', 0).astype(np.float32) / 255

gamma = 0.5
corrected_image = np.power(image, gamma)

cv2.imshow('image', image)
cv2.imshow('corrected_image', corrected_image)
cv2.waitKey()

cv2.imwrite('/tmp/image.png', image*255)
cv2.imwrite('/tmp/corrected_image.png', corrected_image*255)

cv2.destroyAllWindows()
```

Histogram Equalization

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

grey = cv2.imread('../data/Lena.png', 0)
cv2.imshow('original grey', grey)
cv2.waitKey()

hist, bins = np.histogram(grey, 256, [0, 255])
plt.fill(hist)
plt.xlabel('pixel value')
plt.show()

grey_eq = cv2.equalizeHist(grey)
hist, bins = np.histogram(grey_eq, 256, [0, 255])
plt.fill_between(range(256), hist, 0)
plt.xlabel('pixel value')
plt.show()

cv2.imshow('equalized grey', grey_eq)
cv2.waitKey()

color = cv2.imread('../data/Lena.png')
hsv = cv2.cvtColor(color, cv2.COLOR_BGR2HSV)

hsv[..., 2] = cv2.equalizeHist(hsv[..., 2])
color_eq = cv2.cvtColor(hsv, cv2.COLOR_HSV2BGR)
cv2.imshow('original color', color)
cv2.imshow('equalized color', color_eq)
cv2.waitKey()
cv2.destroyAllWindows()
```

Image Filtering

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

image = cv2.imread('../data/Lena.png').astype(np.float32) / 255

noised = (image + 0.2 * np.random.rand(*image.shape).astype(np.float32))
noised = noised.clip(0, 1)
plt.imshow(noised[:,:,[2,1,0]])
plt.show()

gauss_blur = cv2.GaussianBlur(noised, (7, 7), 0)
plt.imshow(gauss_blur[:, :, [2, 1, 0]])
plt.show()

median_blur = cv2.medianBlur((noised * 255).astype(np.uint8), 7)
plt.imshow(median_blur[:, :, [2, 1, 0]])
plt.show()

bilat = cv2.bilateralFilter(noised, -1, 0.3, 10)
plt.imshow(bilat[:, :, [2, 1, 0]])
plt.show()
```

다음의 프로그램을 작성하시오.

- Lena 이미지를 컬러영상으로 읽고 화면에 출력하시오.
- 영상을 흑백으로 변환하고 화면에 출력하시오
  - 흑백으로 변환된 영상에 Histogram Equalization을 적용하고 출력하시오.
  - 흑백으로 변환된 영상에 Gamma Correction을 적용하고 출력하시오.
- 영상을 HSV 컬러 스페이스로 변환하고 각각을 화면에 출력하시오. (각 값을 0부터 255로 정규화)
  - H 채널에 대해서 Median Filter를 적용하고 출력하시오.
  - S 채널에 대해서 Gaussian Filter를 적용하고 출력하시오.
  - V 채널에 대해서 Bilateral Filter를 적용하고 출력하시오.