Lec2 视觉感知与成像

·AIPKU·

人工智能引论实践课 计算机视觉小班

主讲人：刘家瑛

部分讲义来源与致谢

1. Rafael Gonzalez, Richard Woods. Digital Image Processing
2. Avideh Zakhor. UCBerkeley EE 225B: Digital Image Processing
3. James Hays. Brown University CSCI 129: Computational Photography
4. James Tompkin. Brown University CSCI 1290: Computational Photography and Image Manipulation
5. Jitendra Malik. Brown University CS 1430: Introduction to Computer Vision
6. Image pyramid tutorial, pyrtools. https://pyrtools.readthedocs.io/en/latest/tutorials/tutorial2.html
7. Golan Levin. Image Processing and Computer Vision
8. 科普文章：颜色只是人类的“发明”？如果人类不存在，那宇宙还有颜色吗?

What is a IMAGE

Image & Pixel

- Each part of an **image** is a **pixel**

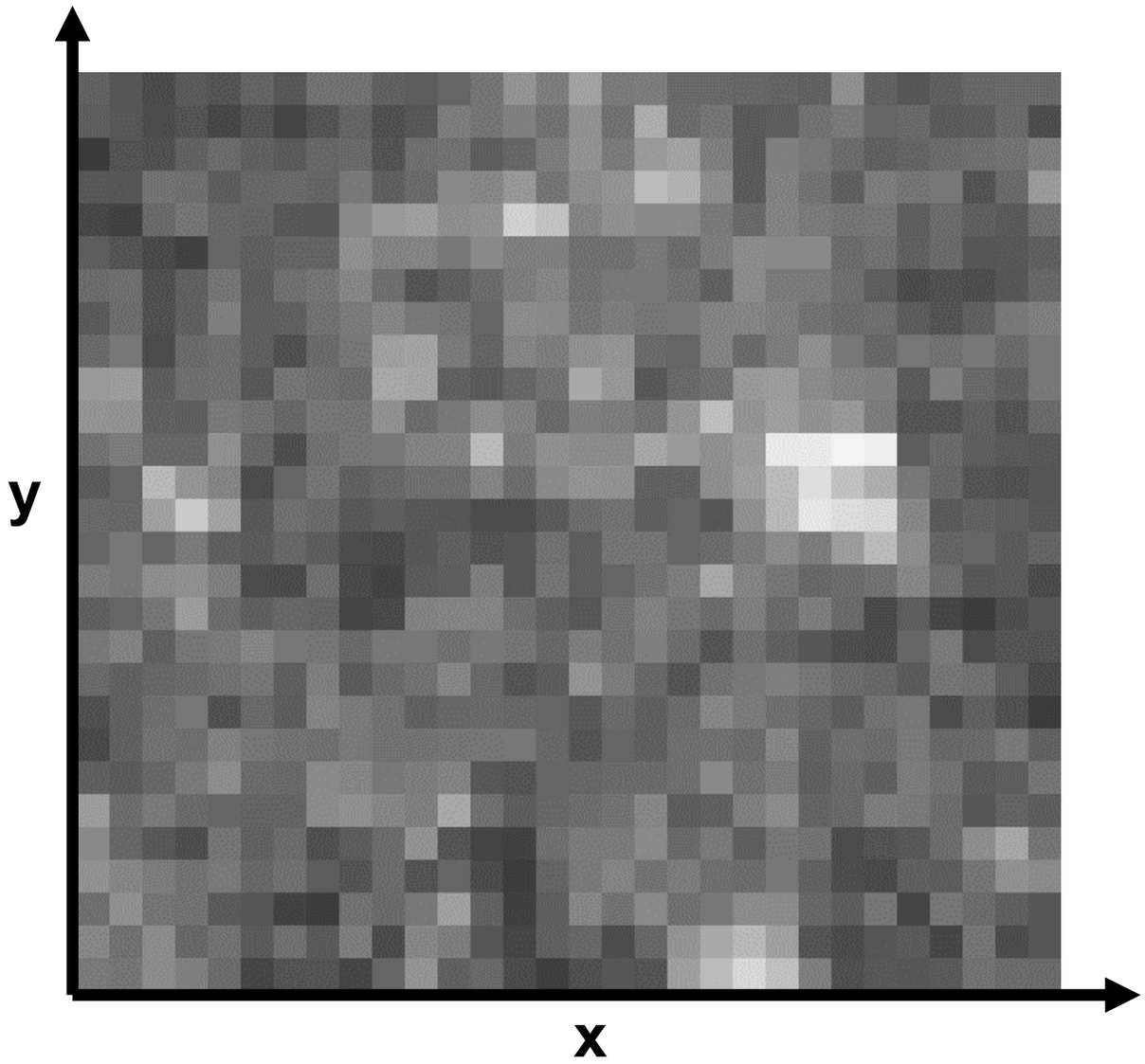
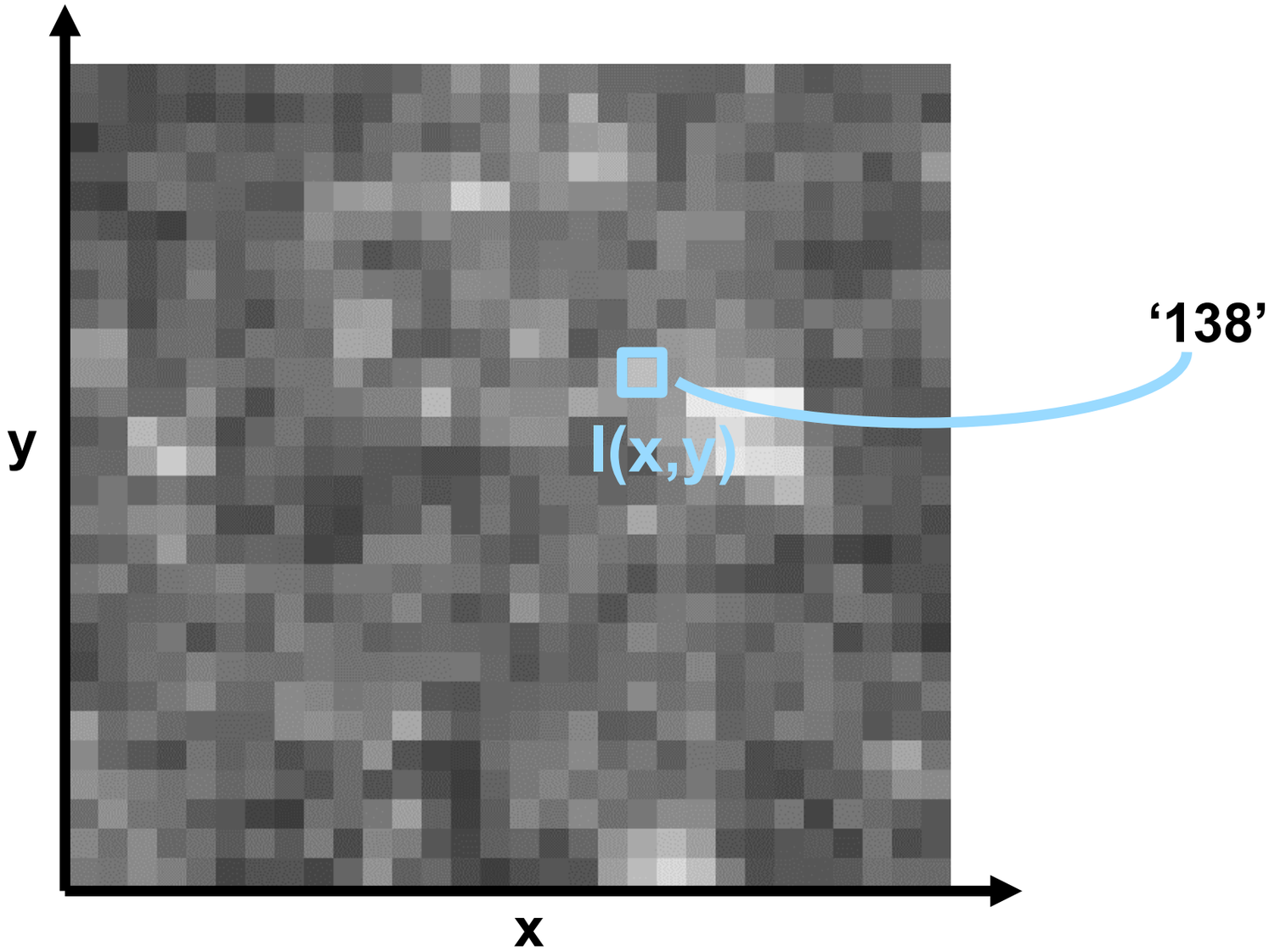


Image & Pixel

- Each part of an image is a pixel
  - Pixel → Picture Element



Source: James Tompkin

Image as a 2D Sampling of Signal

- **Signal:**

  Function depends on some variables with physical meaning

- **Image:** Sampling of that the function

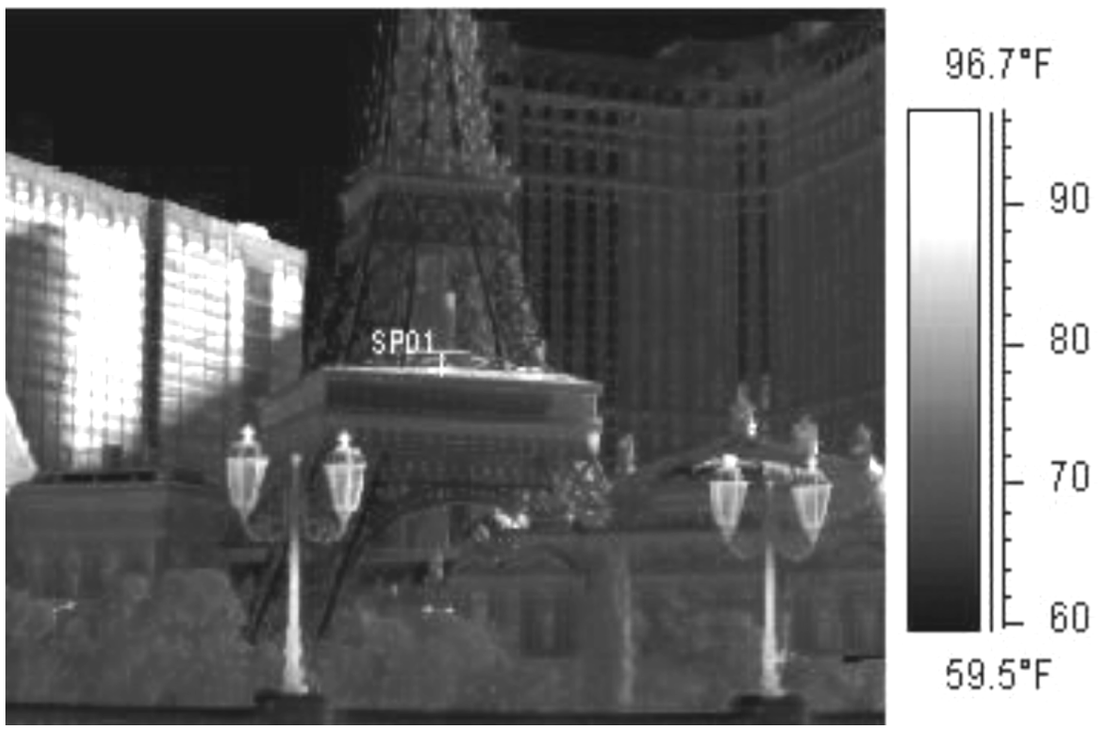
  - 2 variables: x-y coordinates
  - 3 variables: x-y + $time$ (video)
  - ‘Brightness’ is the value of the function for visible light

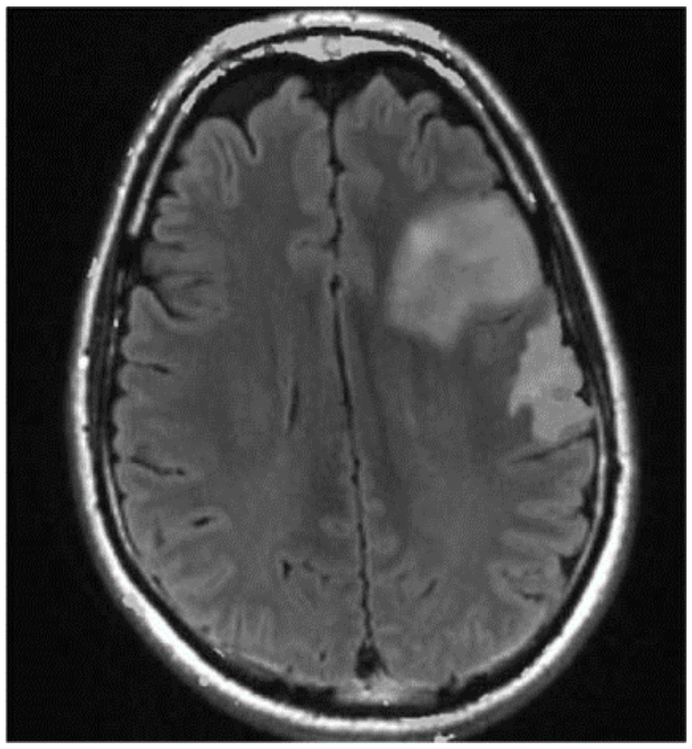
- Can be other physical values as well:

  Temperature, Pressure, Depth, ...

Source: James Tompkin

Example 2D Images



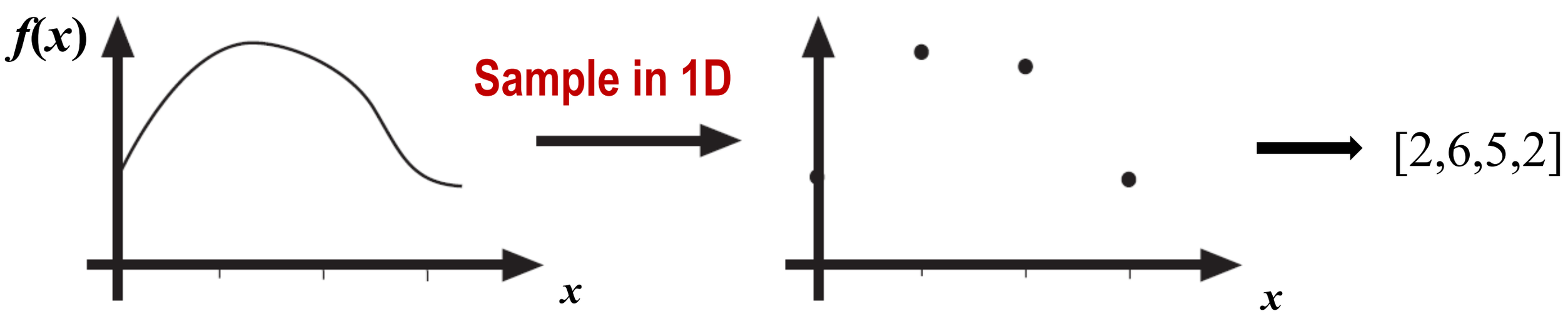


Source: James Tompkin

Sampling in 1D

- **Sampling in 1D** takes a ‘function’,

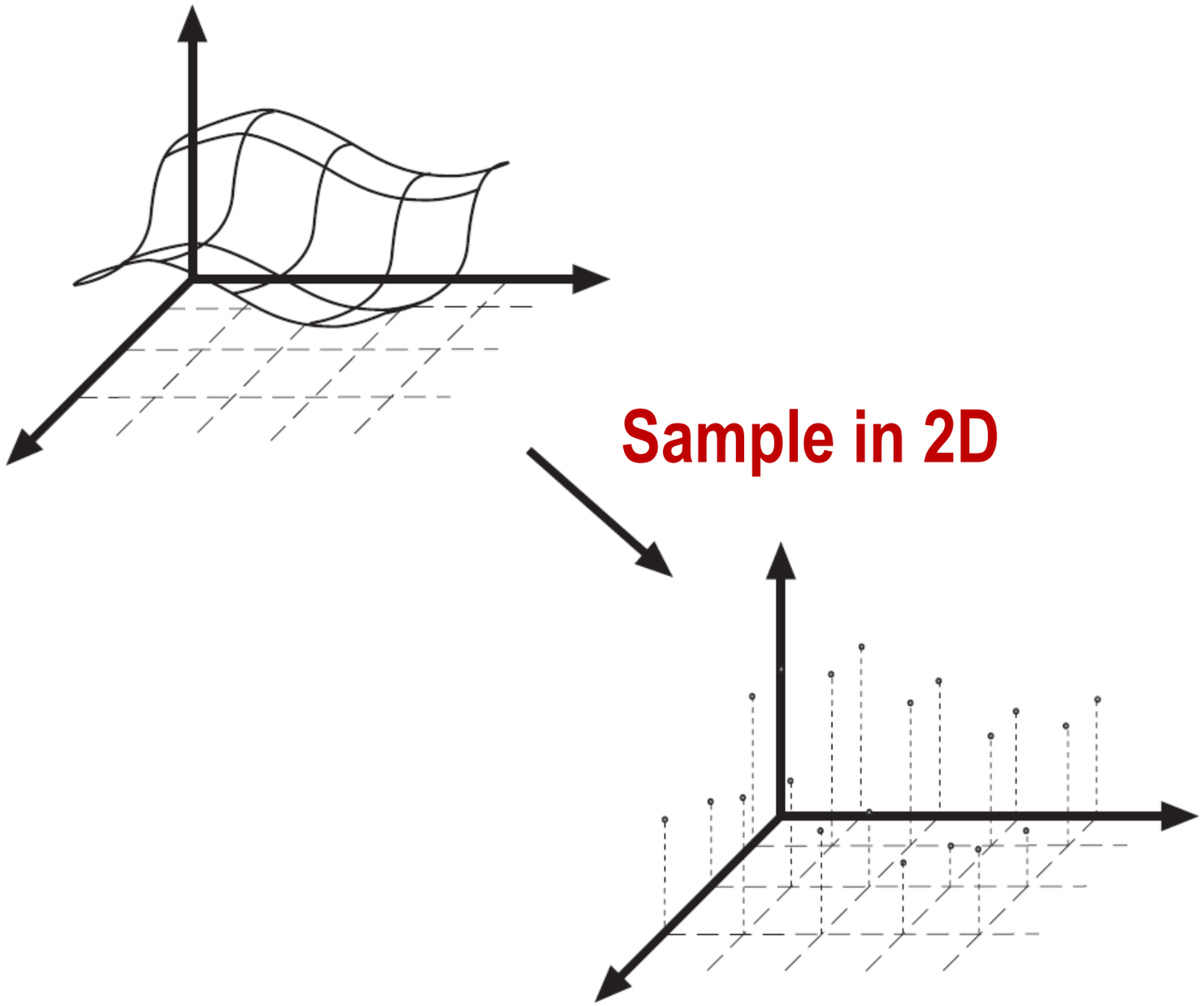
and returns a vector whose elements are values of that function at the sample points.



Source: Danny Alexander

Sampling in 2D

- **Sampling in 2D** takes a function and returns a matrix.



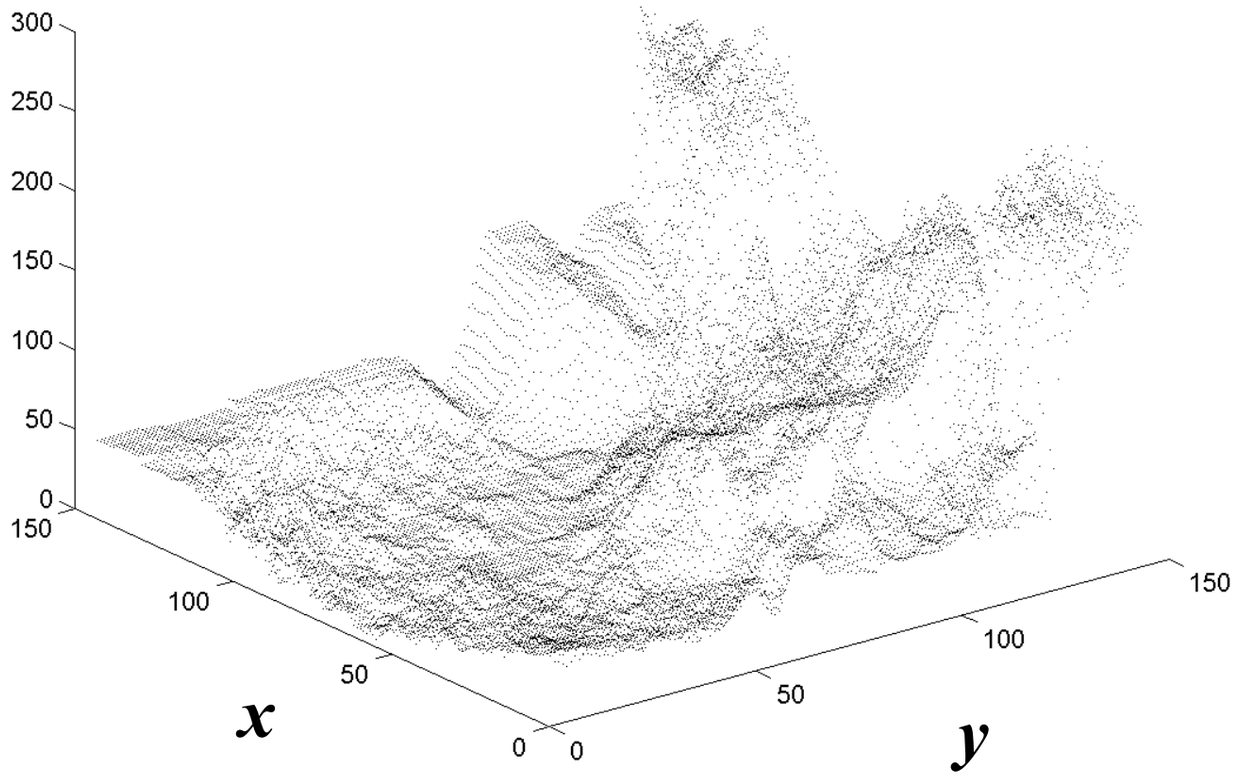
Source: Danny Alexander

Grayscale Image

- **Grayscale Digital Image** as ‘Height Map’

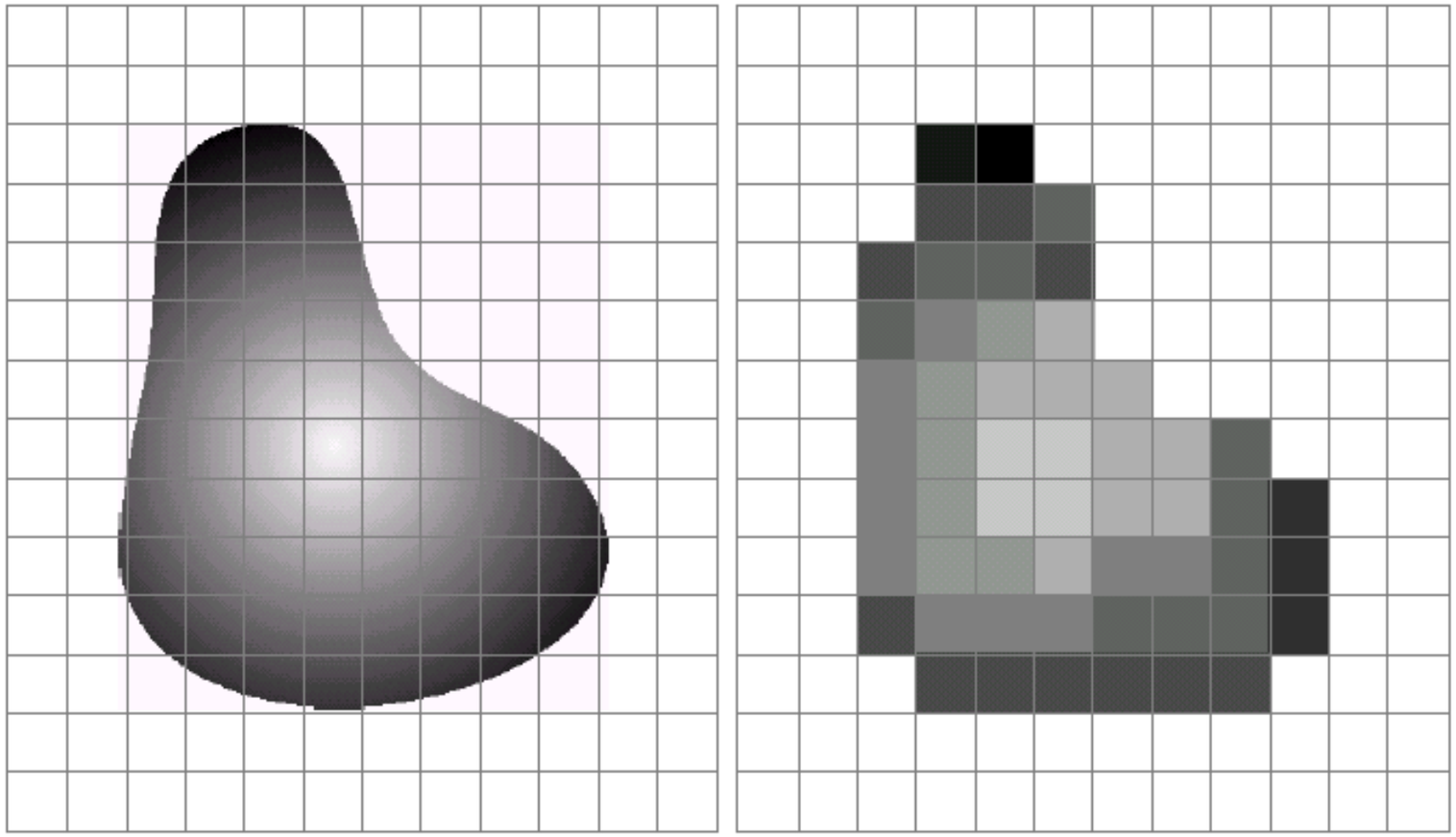
Brightness

Intensity I



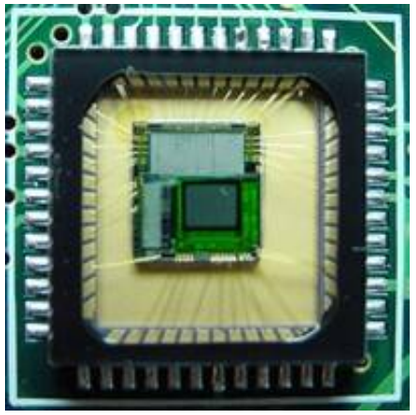
Source: Danny Alexander

Sensor Array



a b

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



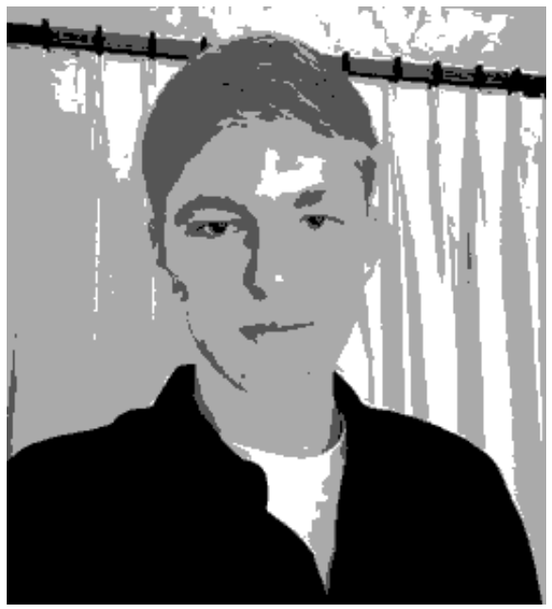
CMOS sensor

Quantization

- **Quantization Effects** – Radiometric Resolution

8 bit
256 levels

4 bit
16 levels



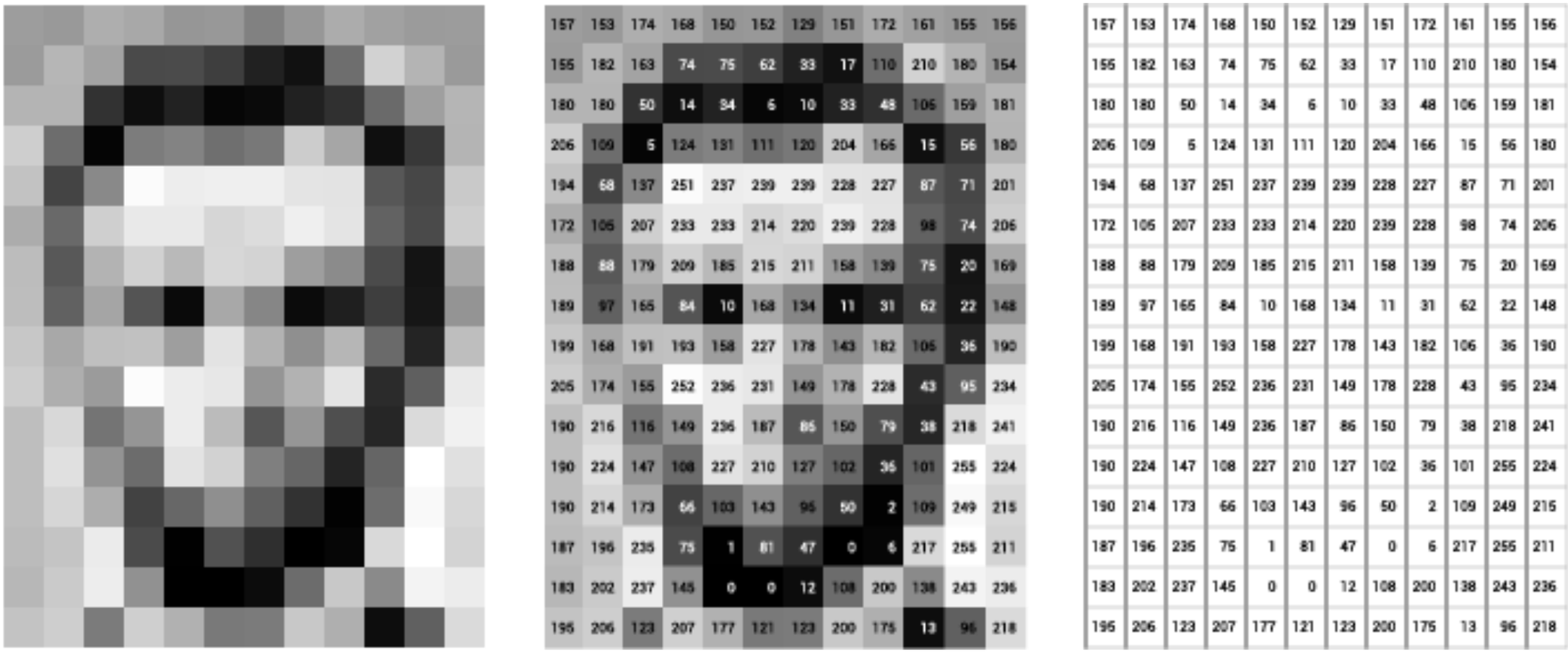
2 bit
4 levels

1 bit
2 levels

Source: James Tompkin

Image Basics

- A grayscale image ‘Abraham Lincoln’
  - Each pixel's brightness: a single 8-bit number
  - Range is from **0 (black) to 255 (white)**



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

Pixel data diagram. At left, an image of Lincoln; at center, the pixels labeled with numbers from 0-255, representing their brightness; and at right, these numbers by themselves.

Source: Openframeworks

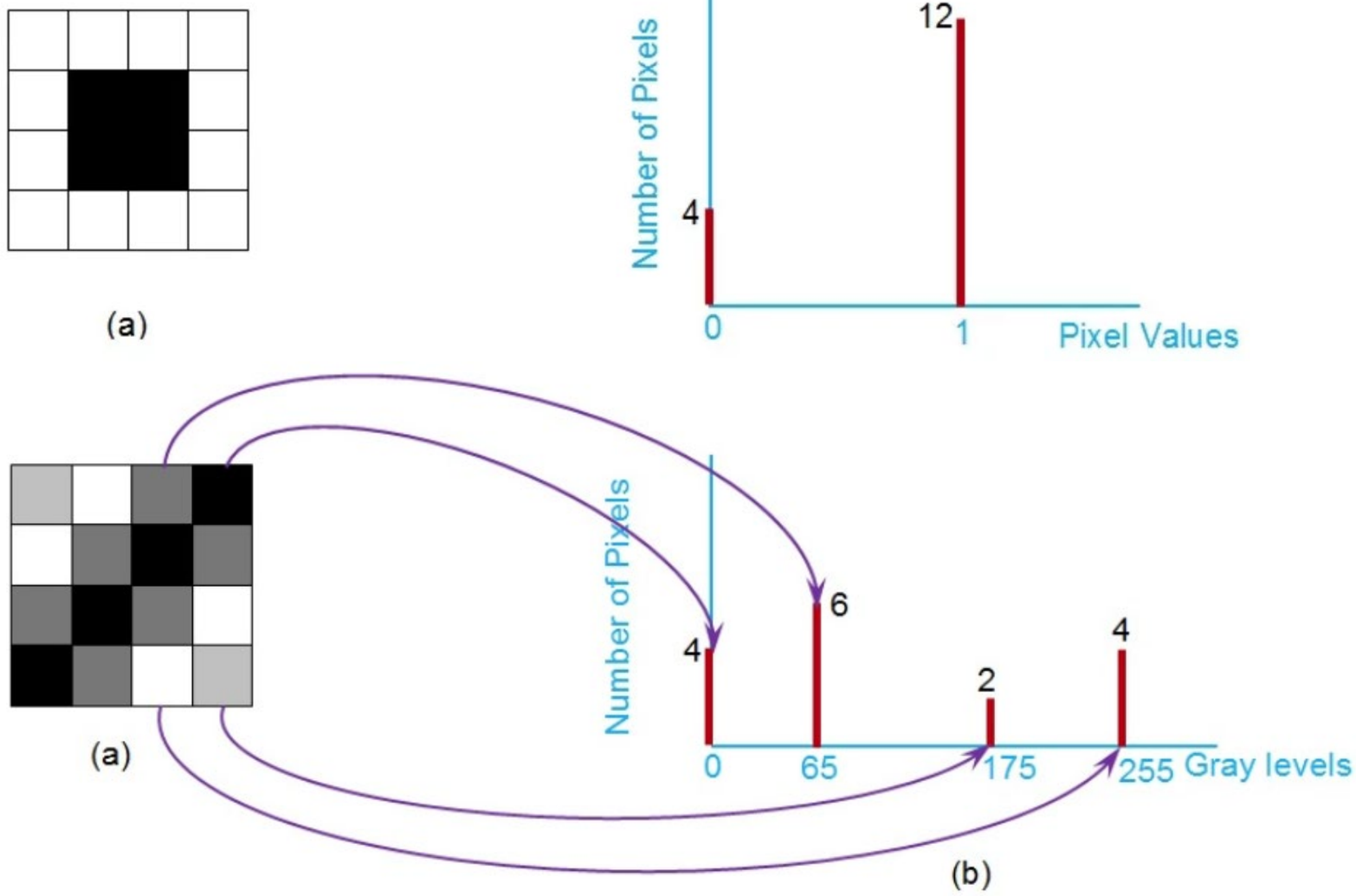
Image Basics

```
{157, 153, 174, 168, 150, 152, 129, 151, 172, 161, 155, 156,
 155, 182, 163,  74,  75,  62,  33,  17, 110, 210, 180, 154,
 180, 180,  50,  14,  34,   6,  10,  33,  48, 106, 159, 181,
 206, 109,   5, 124, 131, 111, 120, 204, 166,  15,  56, 180,
 194,  68, 137, 251, 237, 239, 239, 228, 227,  87,  71, 201,
 172, 105, 207, 233, 233, 214, 220, 239, 228,  98,  74, 206,
 188,  88, 179, 209, 185, 215, 211, 158, 139,  75,  20, 169,
 189,  97, 165,  84,  10, 168, 134,  11,  31,  62,  22, 148,
 199, 168, 191, 193, 158, 227, 178, 143, 182, 106,  36, 190,
 205, 174, 155, 252, 236, 231, 149, 178, 228,  43,  95, 234,
 190, 216, 116, 149, 236, 187,  86, 150,  79,  38, 218, 241,
 190, 224, 147, 108, 227, 210, 127, 102,  36, 101, 255, 224,
 190, 214, 173,  66, 103, 143,  96,  50,   2, 109, 249, 215,
 187, 196, 235,  75,   1,  81,  47,   0,   6, 217, 255, 211,
 183, 202, 237, 145,   0,   0,  12, 108, 200, 138, 243, 236,
 195, 206, 123, 207, 177, 121, 123, 200, 175,  13,  96, 218};
```

Pixel Intensity Histogram

- **Histogram**

An image histogram is a graph of pixel intensity (on the x-axis) versus number of pixels (on the y-axis)



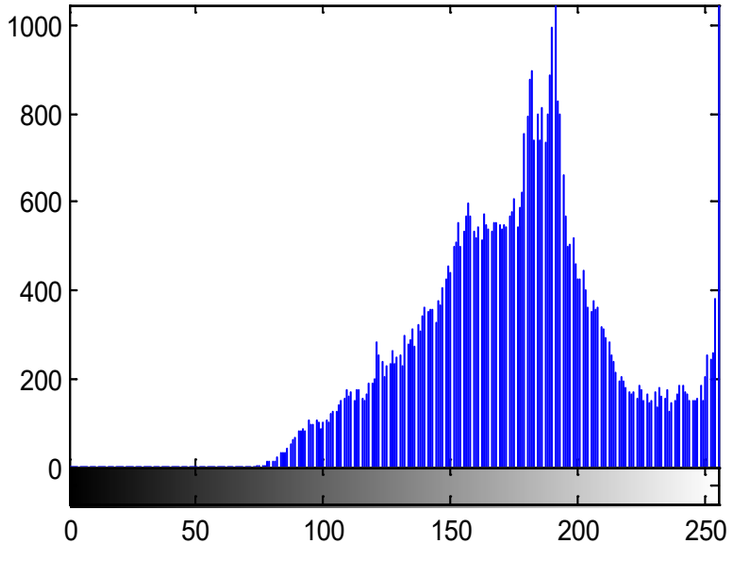
Pixel Intensity Histogram

- **Analyzing Histograms**

- The total number of pixels

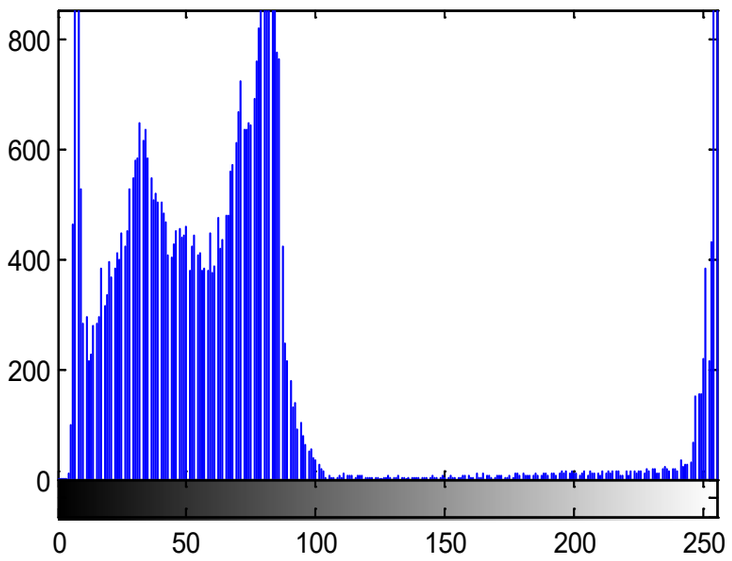
- Image brightness

- Contrast of the image



- **Drawback**

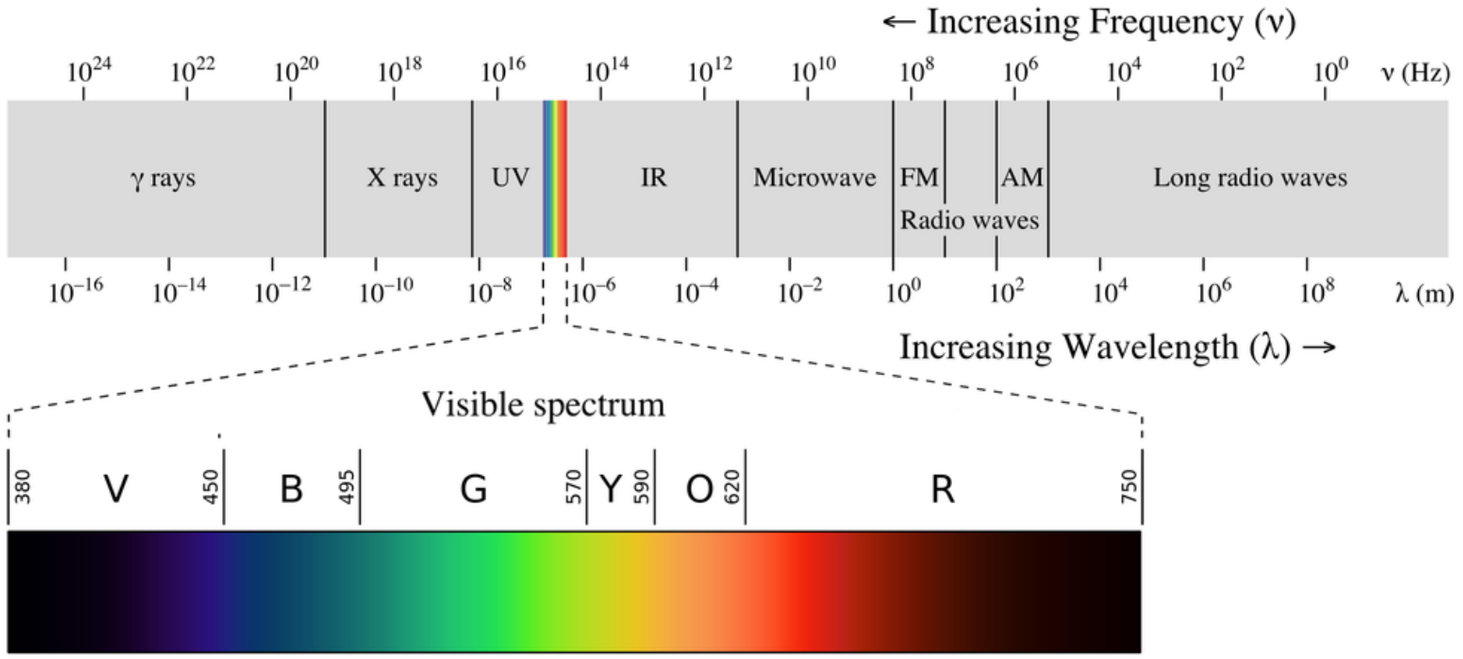
No information regarding the spatial distribution



- # What is COLOR

光与颜色

- 白色光是一种**复合光**
- 光是一种电磁波，它有不同的频率或者波长，不同的波长就代表了不同的颜色
- 在现实世界中，眼睛只能看到一小部分的波长范围，这个范围就是**可见光谱**



波长范围在380nm到750nm之间

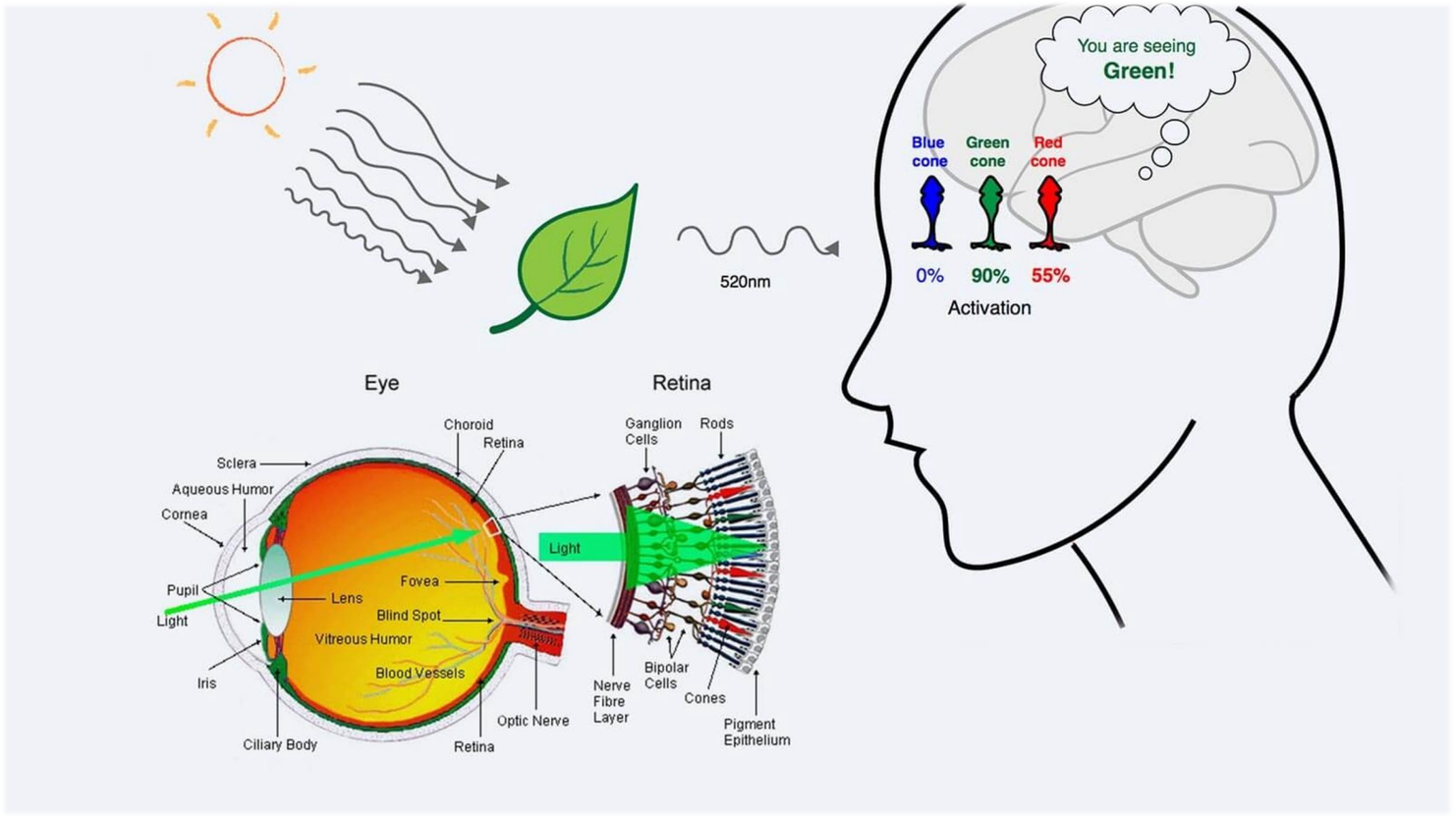
光与颜色

- 颜色不是宇宙的本质属性，其实并不存在，只是在生物的感官中，不同波长的光才表现出了颜色
- 在我们的眼睛里，520nm波长的光经过大脑处理以后，我们将这个波长的光解析为绿色

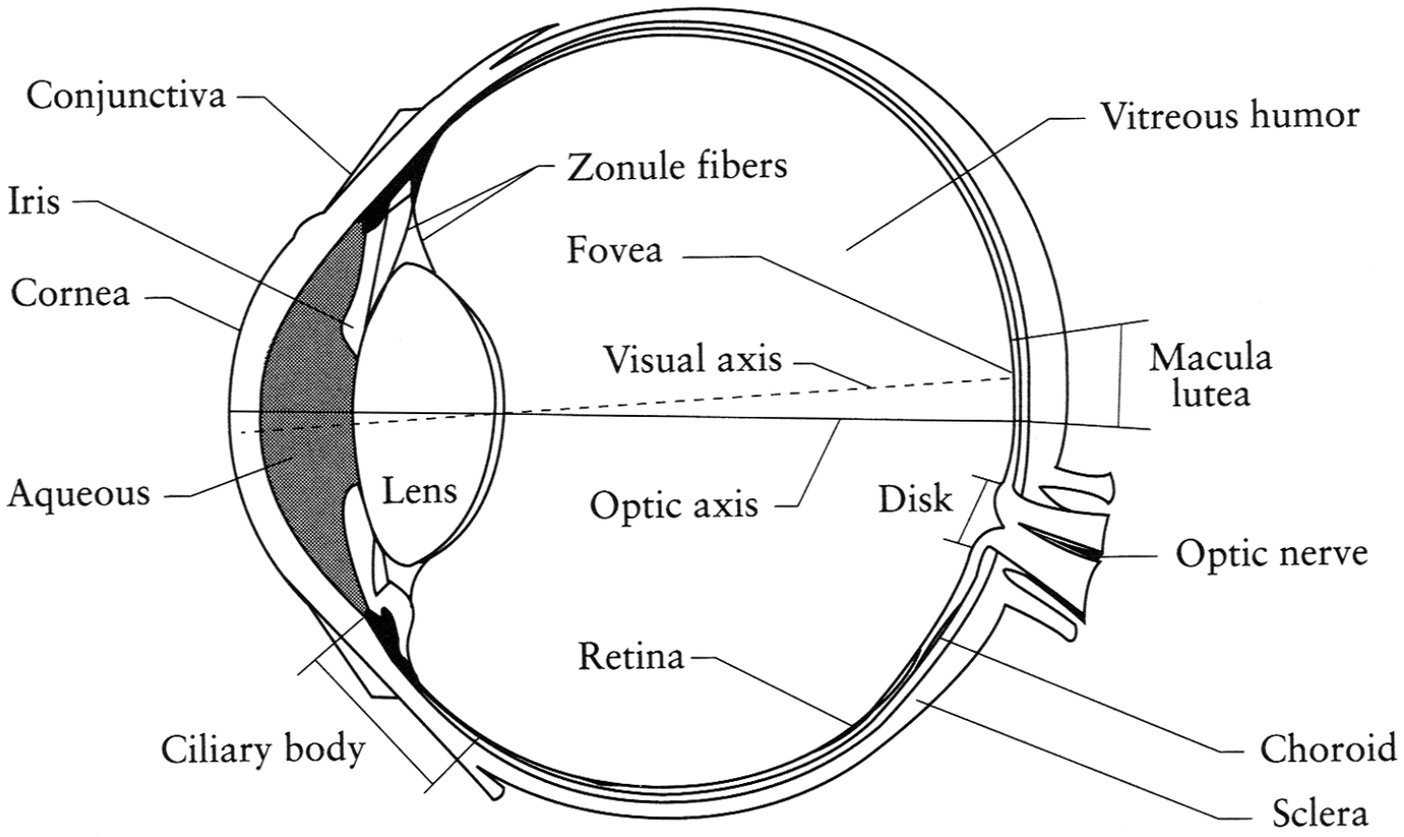
- 我们眼睛所看到的光子，来源有可能直接是光源发出的，如太阳、恒星、黑体辐射、或者化学反应，也可能是被其他物体反射出来的

光与颜色

- 当光子进入我们的眼睛时，首先会通过晶状体然后达到视网膜
- 在视网膜上有两种对光线敏感的细胞，分别是视杆细胞和视锥细胞



The Eye

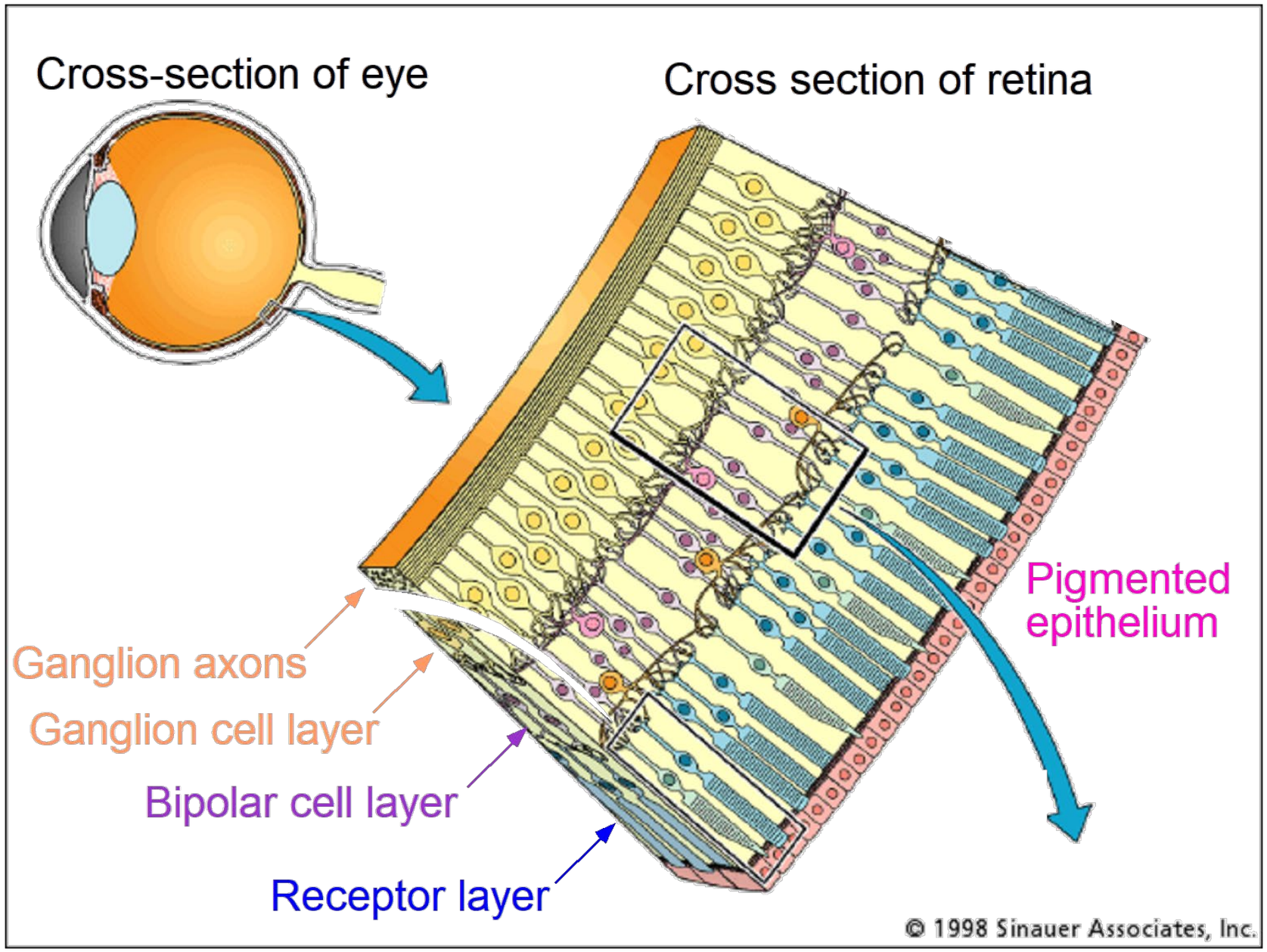


- **The human eye is a CAMERA!**
  - **Iris (虹膜)** - Colored annulus with radial muscles
  - **Pupil (瞳孔)** - Hole (aperture) whose size is controlled by the iris
  - What's the "film"?

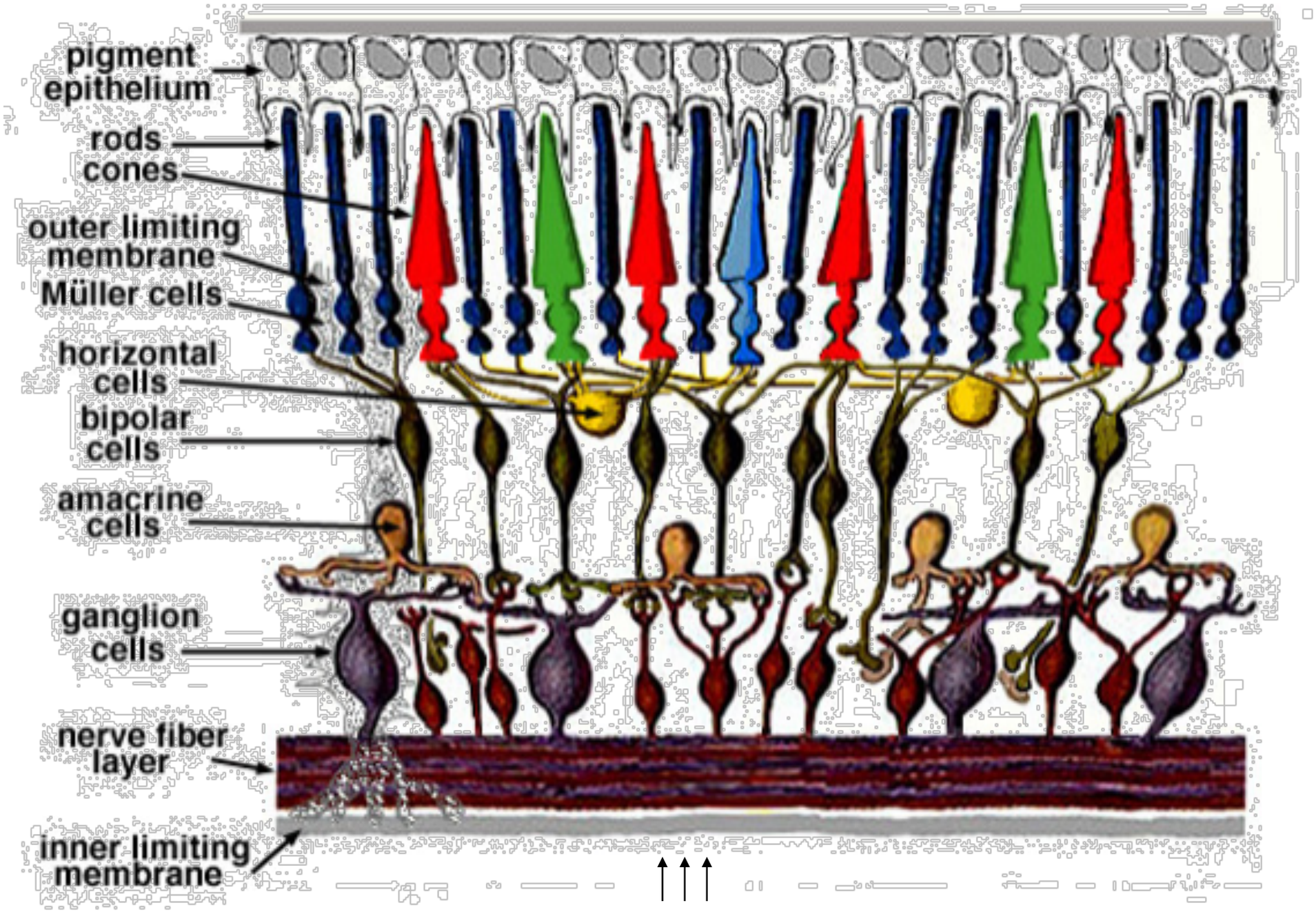
    -- Photoreceptor cells (rods and cones) in the **Retina**

Source: Steve Seitz

The Retina


Cross-section of eye
Cross section of retina
Pigmented epithelium
Ganglion axons
Ganglion cell layer
Bipolar cell layer
Receptor layer
© 1998 Sinauer Associates, Inc.

Retina Up-Close



Rod / Cone

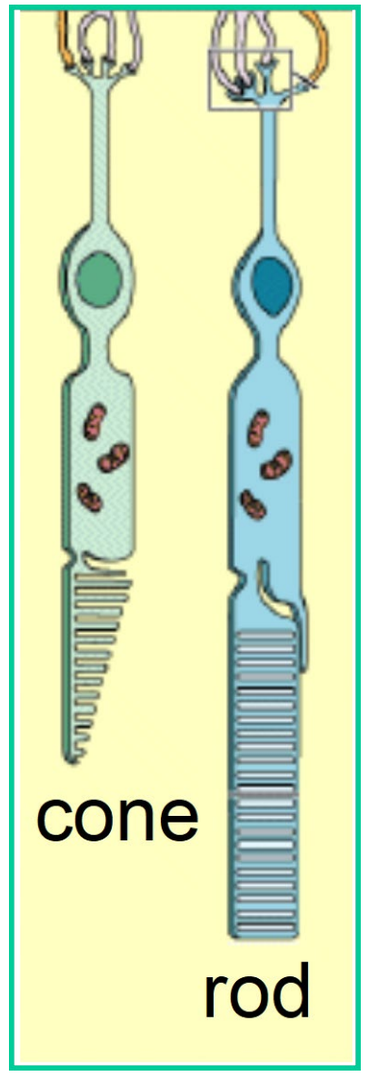
- **Two types of light-sensitive receptors**

Cones

- cone-shaped
- less sensitive
- operate in high light
- color vision

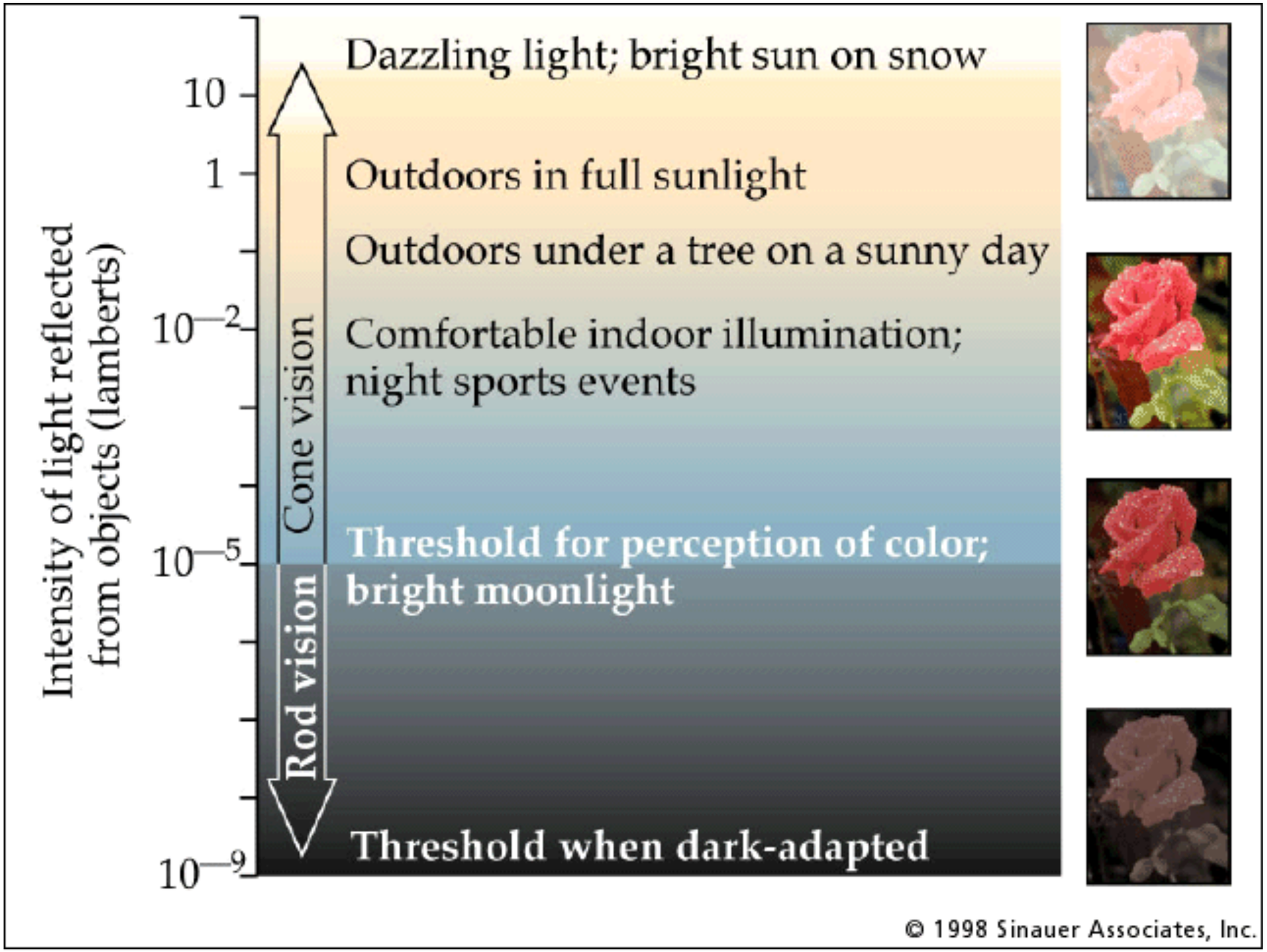
Rods

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



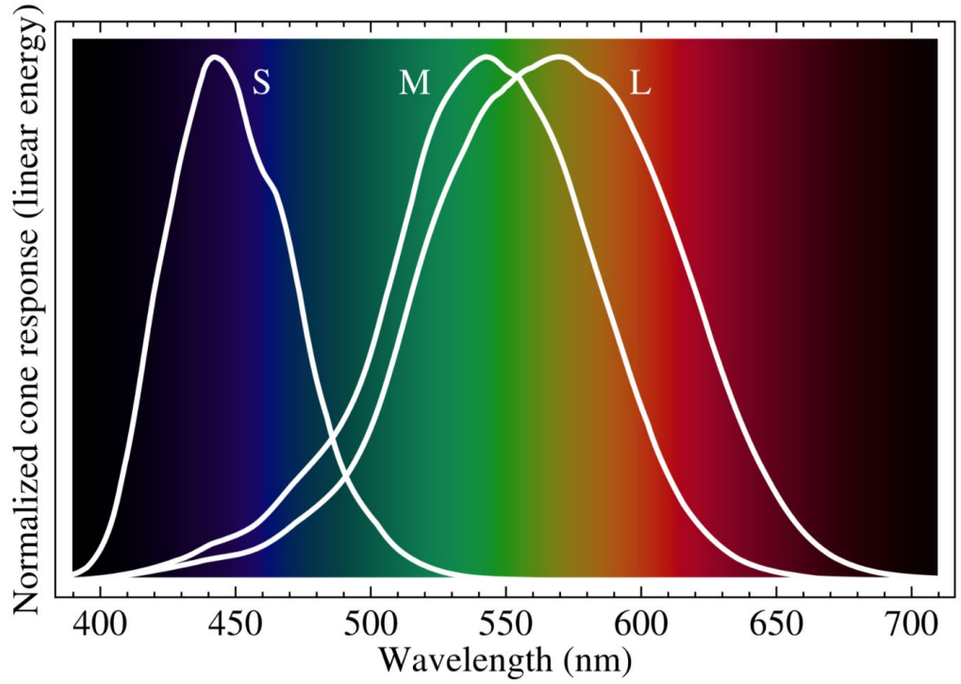
Source: James Tompkin

Rod / Cone Sensitivity

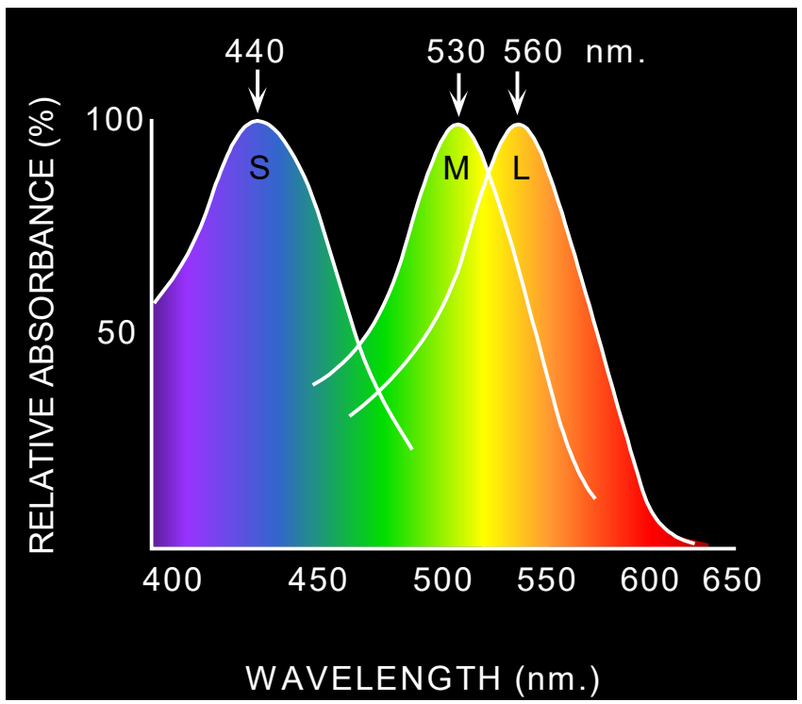


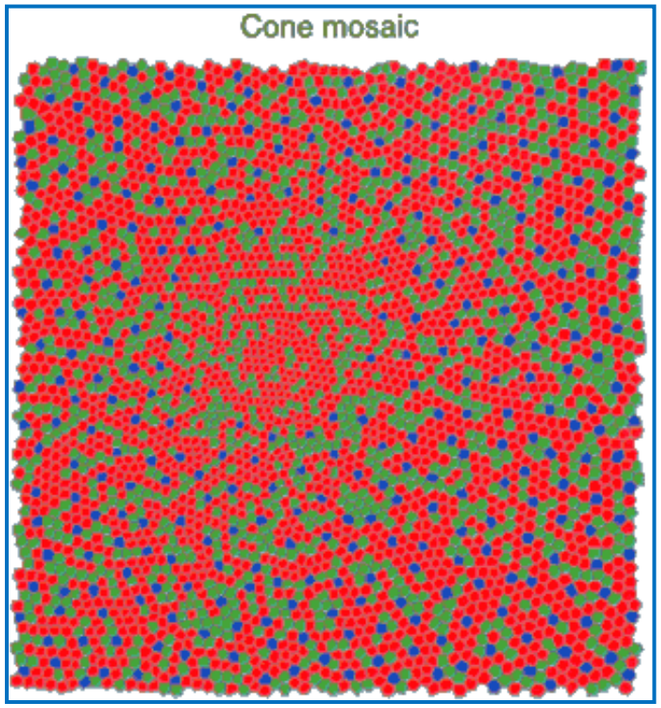
Electromagnetic Spectrum

Human Luminance Sensitivity Function



Three kinds of cones:

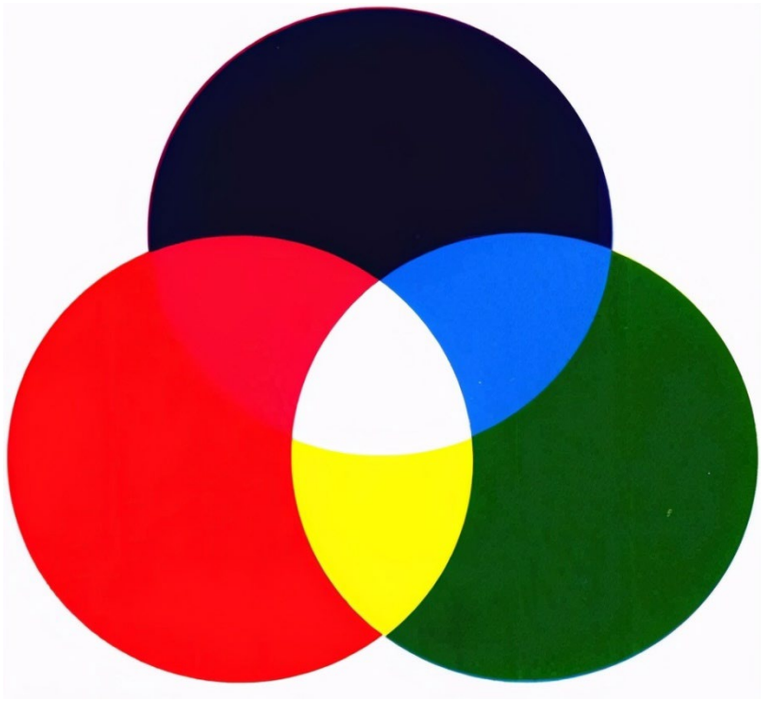




http://www.yorku.ca/eye/photopik.htm

光与颜色

- 传播进眼睛的光线大部分情况下是复合光，当三种视锥细胞被同时点亮时，大脑就会认为这是白色；红色和绿色就是黄色，绿色和蓝色就是青色
- 当然其他的颜色颜色会根据视锥细胞被点亮的程度给达到传递信号，大脑经过分析就可以得出其他颜色
- 所以一个正常人它就能大约分辨出大约**100万种不同的颜色**

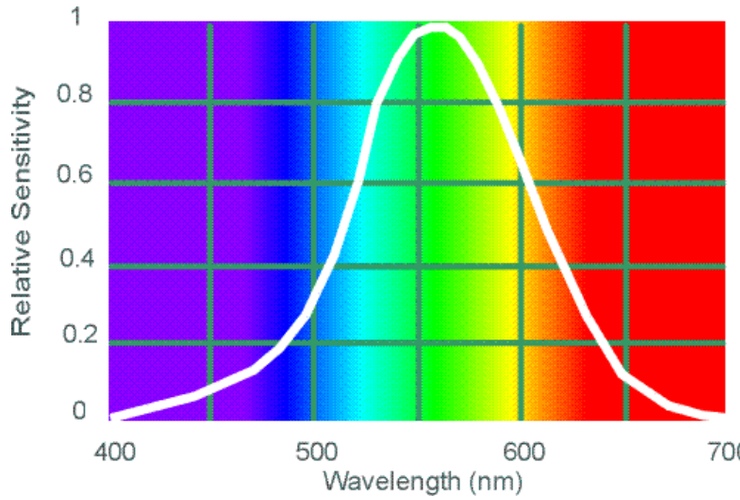


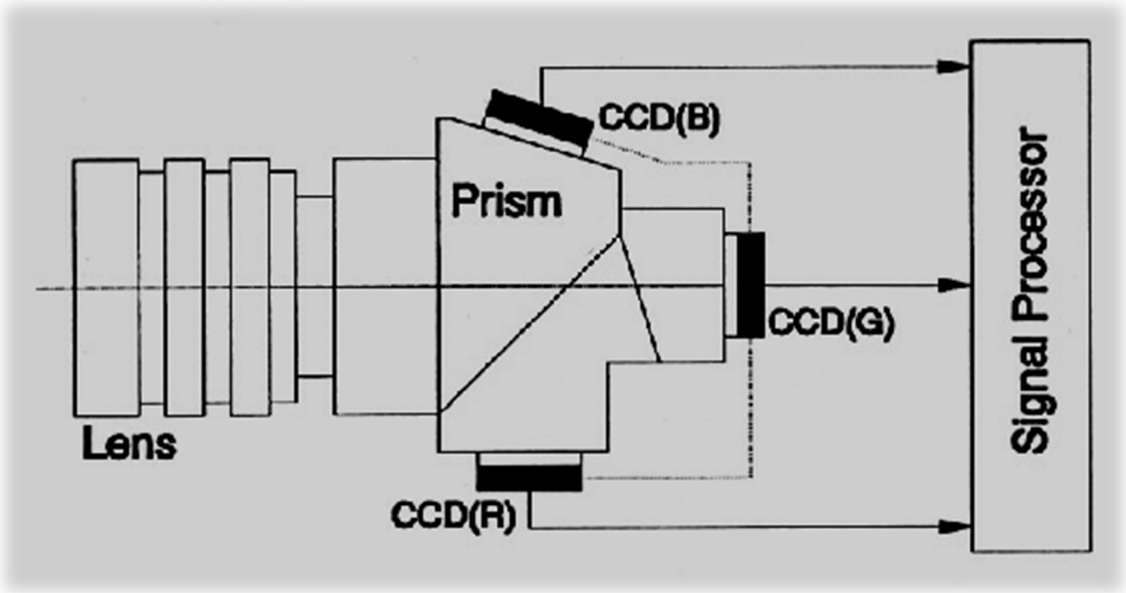
光与颜色

- **狗**的眼睛中只有**两种视锥细胞**，一种辨别蓝色区域，一种辨别黄色区域；但是狗的眼睛中有更加**丰富的视杆细胞**，这使得它们可以夜晚**微弱的光线**下看清楚
- 鸟类的眼睛就更为先进，不仅有着更好的视觉，也有着非常强的辨别颜色的能力，它们有**四种视锥细胞**，其中一种可以使得它们对紫外
- 在自然界中，最为神奇的眼睛是螳螂虾，它们拥有12种视锥细胞，能够分辨10^32种颜色线波段的光线敏感，所以它们可以看到紫外线

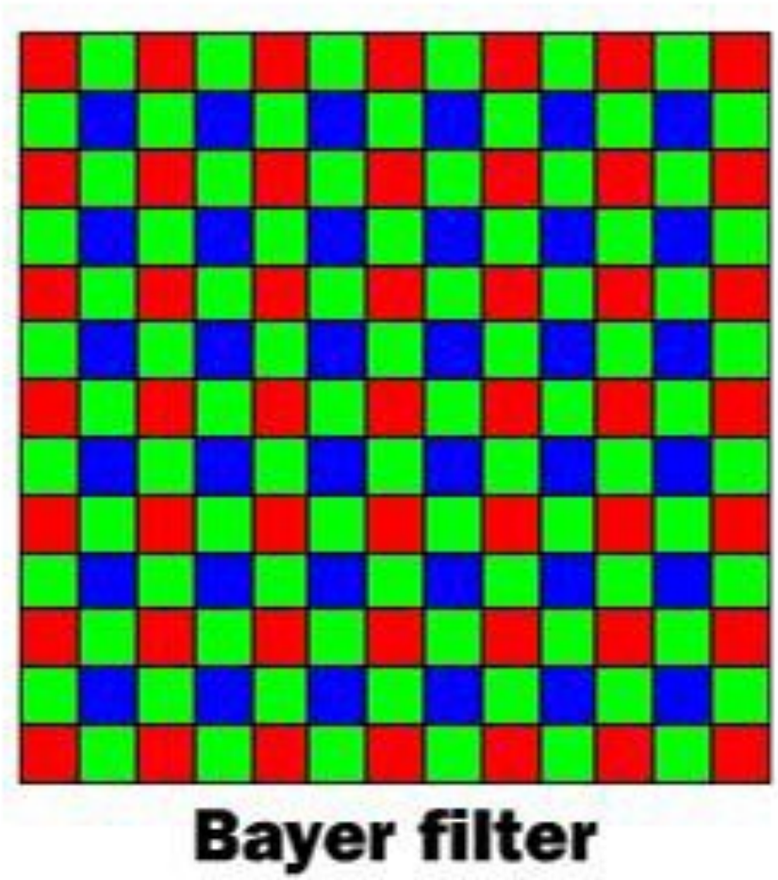
Color Sensing in Camera (RGB)

- 3-Chip *Vs*. 1-Chip : Quality *Vs*. Cost
- Why more green?





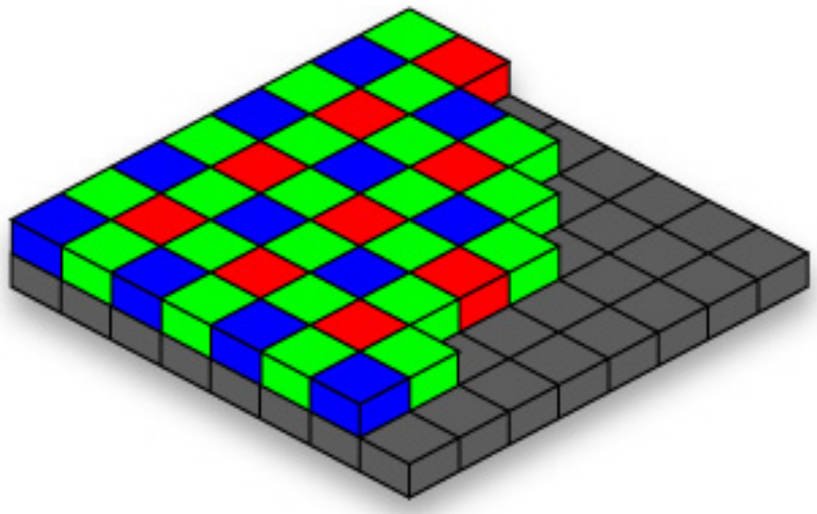
Why 3 colors?

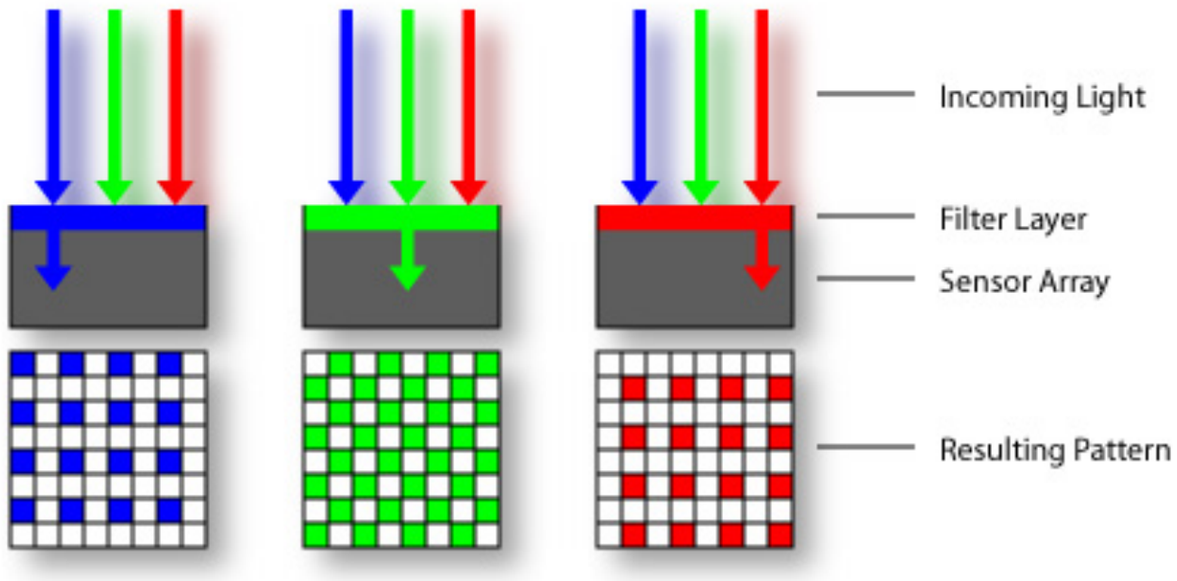


http://www.cooldictionary.com/words/Bayer-filter.wikipedia

Practical Color Sensing: Bayer Grid

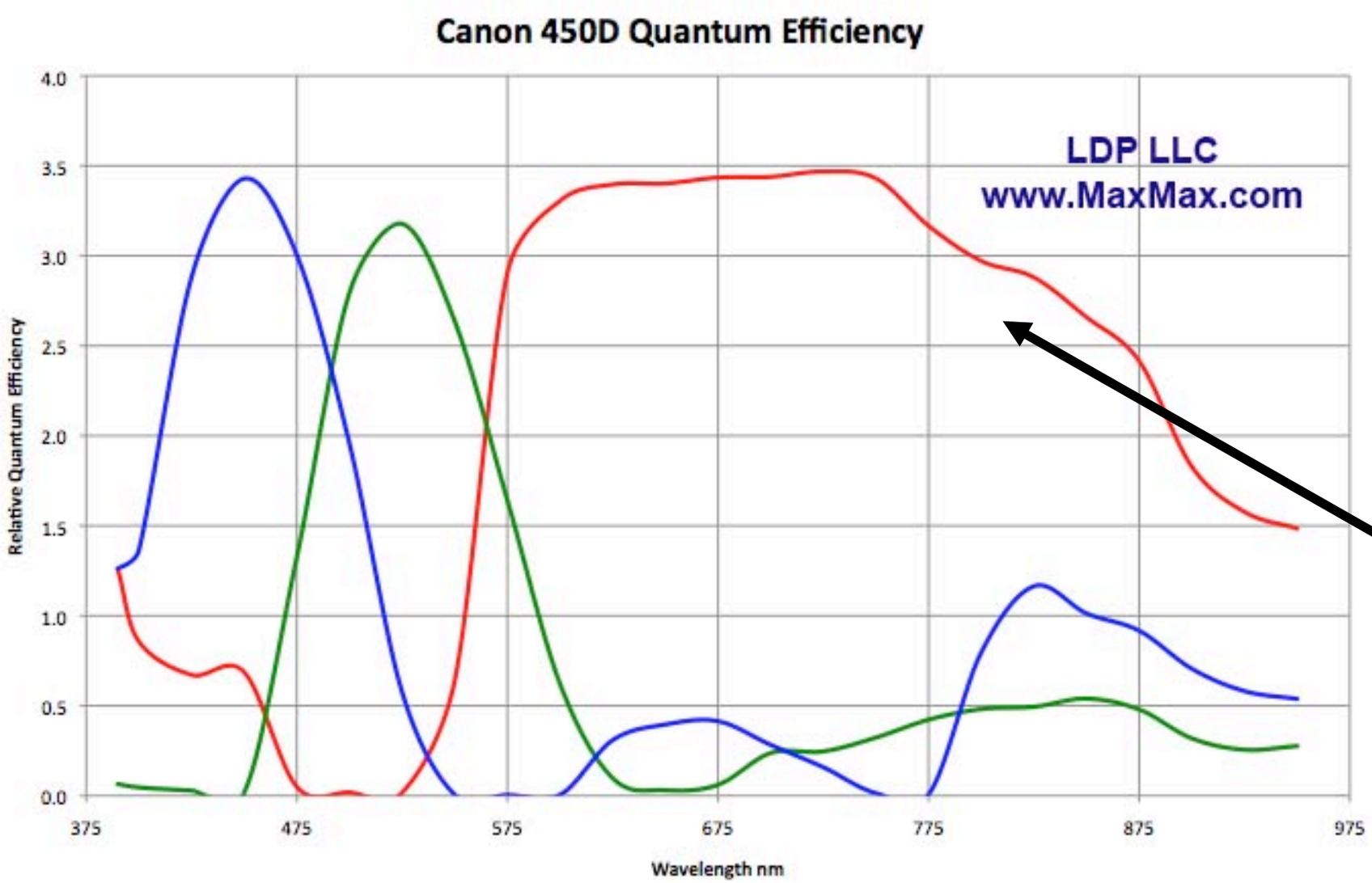
- **Estimate RGB**
  at ‘G’ cells from neighboring values





Source: Steve Seitz

Camera Color Response

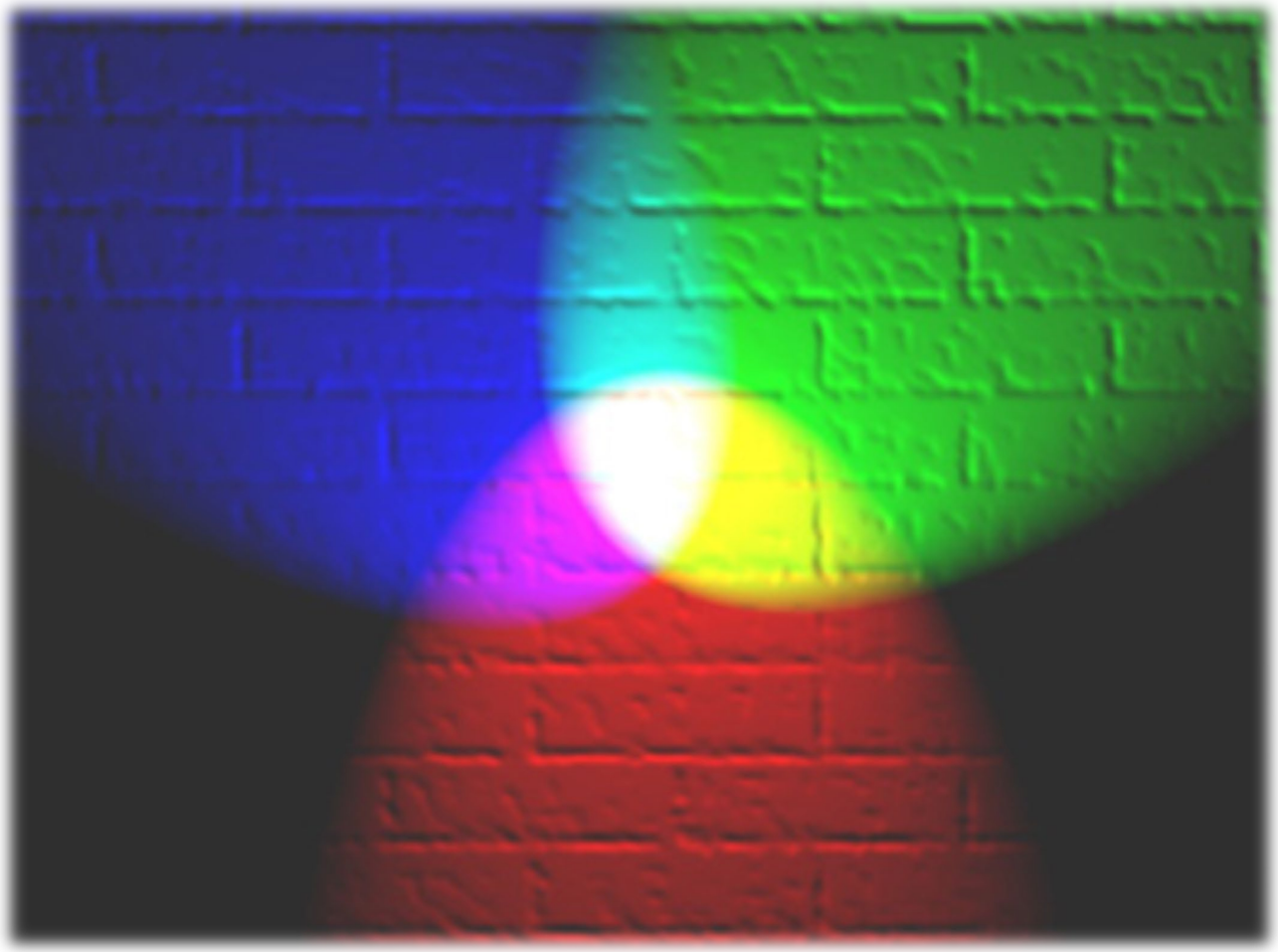


Canon 450D

What's going on over here?

Color Spaces

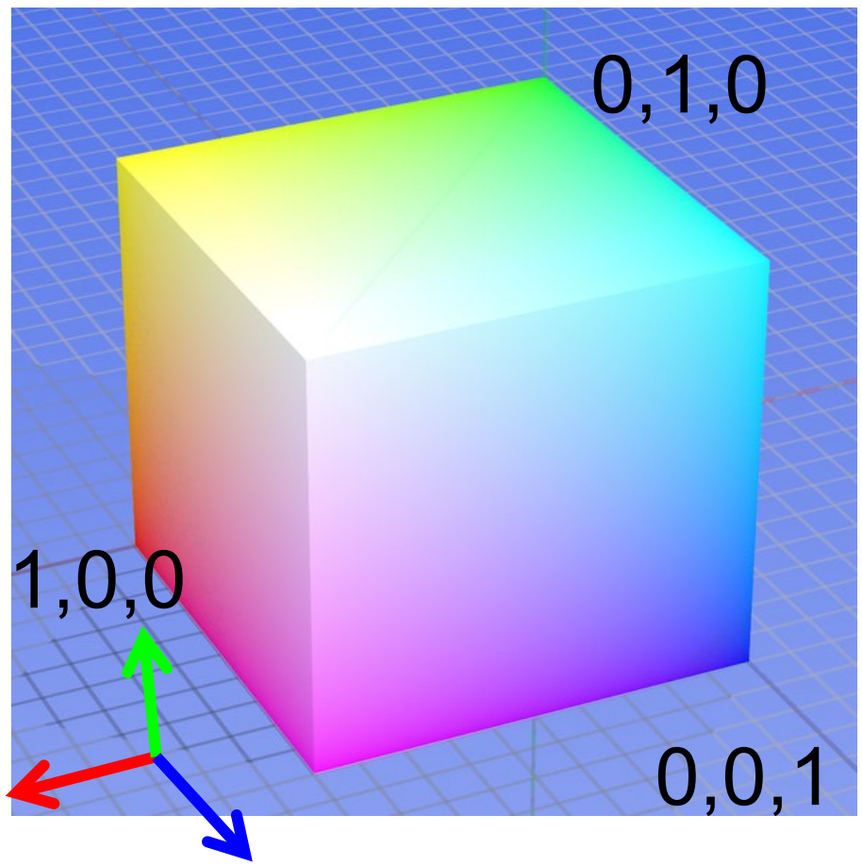
- How can we represent color?



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

Color Spaces: RGB

- **Default Color Space**



R
(G=0,B=0)

G
(R=0,B=0)

B
(R=0,G=0)

- Drawbacks
  - Strongly correlated channels
  - Non-perceptual

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

If you had to choose, would you rather go without:

- intensity (‘value’), or
- hue + saturation (‘chroma’)?

Most Information in Intensity

Only color shown – constant intensity

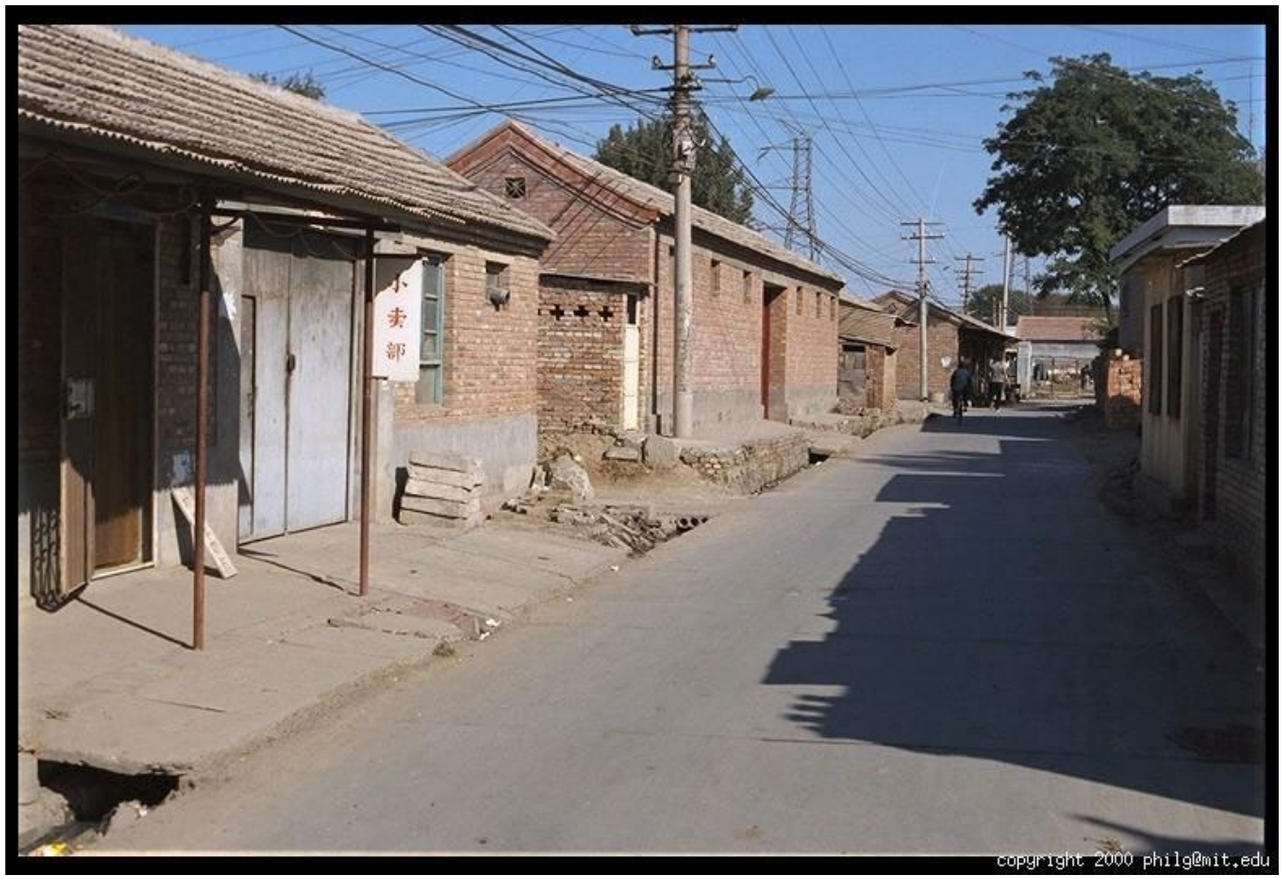
Source: James Hays

Most Information in Intensity

Only intensity shown – constant color

Source: James Hays

Most Information in Intensity

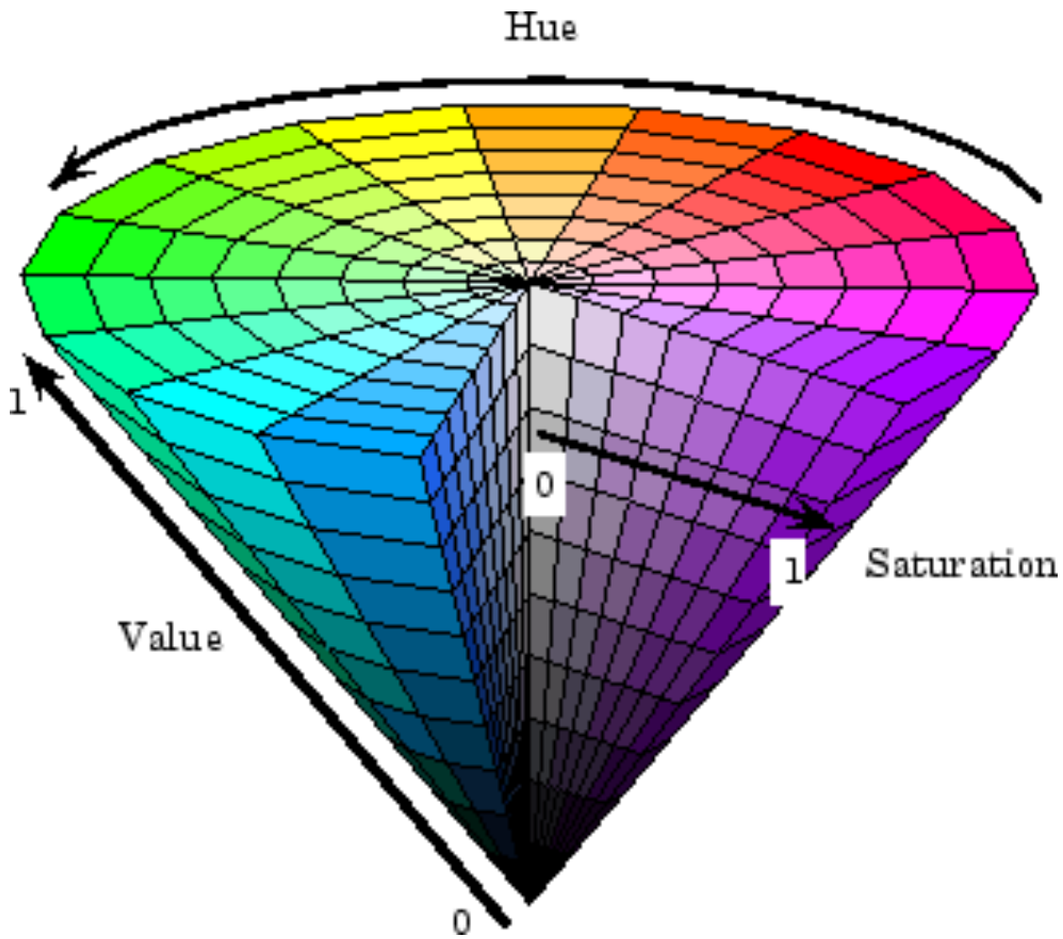


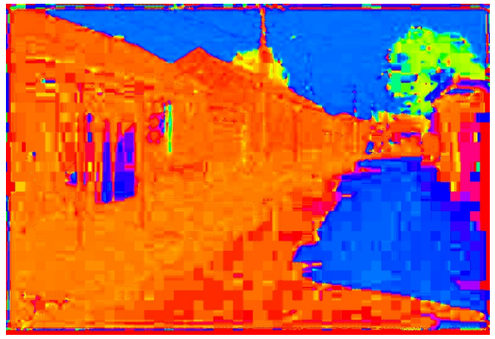
Original image

Source: James Hays

Color Spaces: HSV

Intuitive color space





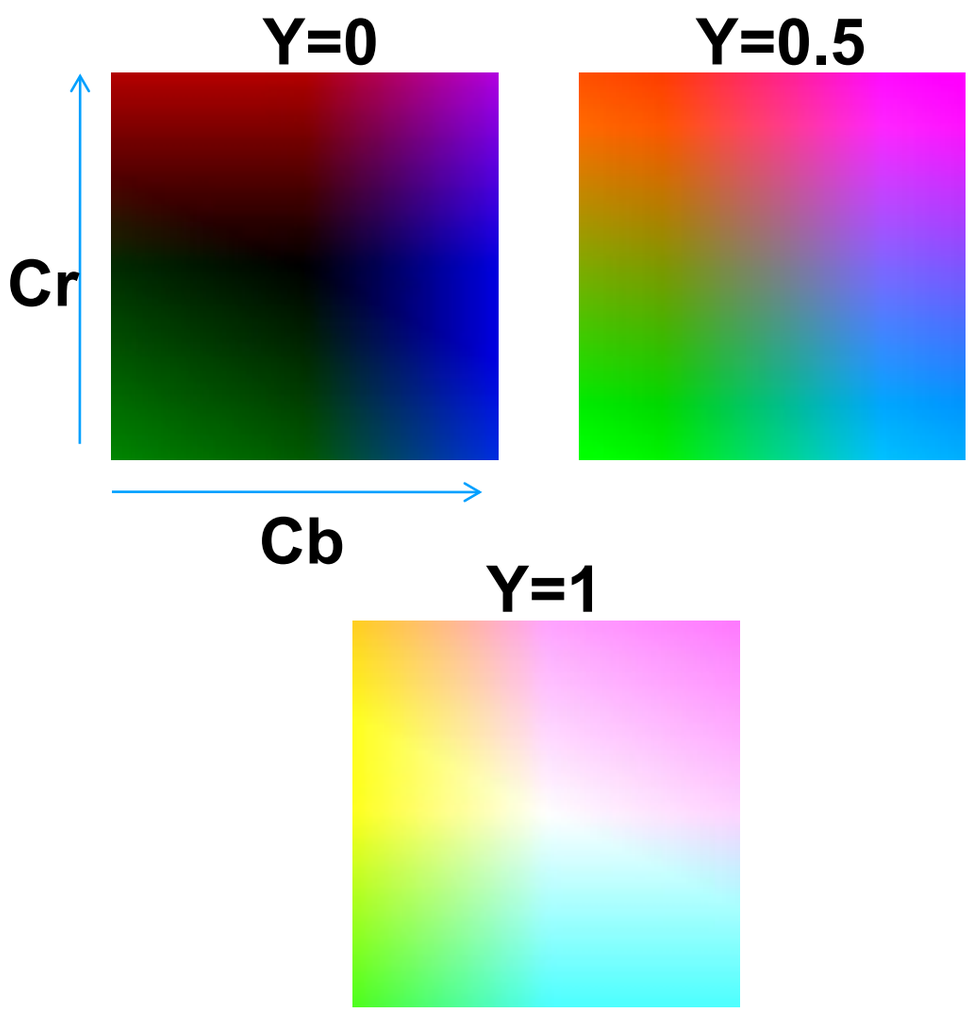
H
(S=1,V=1)

S
(H=1,V=1)

V
(H=1,S=0)

Color Spaces: YCbCr

Fast to compute, good for compression, used by TV



Y
(Cb=0.5,Cr=0.5)

Cb
(Y=0.5,Cr=0.5)

Cr
(Y=0.5,Cb=05)

Subsample Chroma

Most JPEG images & videos
subsample chroma

PSP Comp 3
2x2 Chroma subsampling
285K

Original
1,261K lossless
968K PNG

Source: James Tompkin

Lec2 Image Transformation

AIPKU

人工智能引论实践课 计算机视觉小班

主讲人：刘家瑛

Transformation

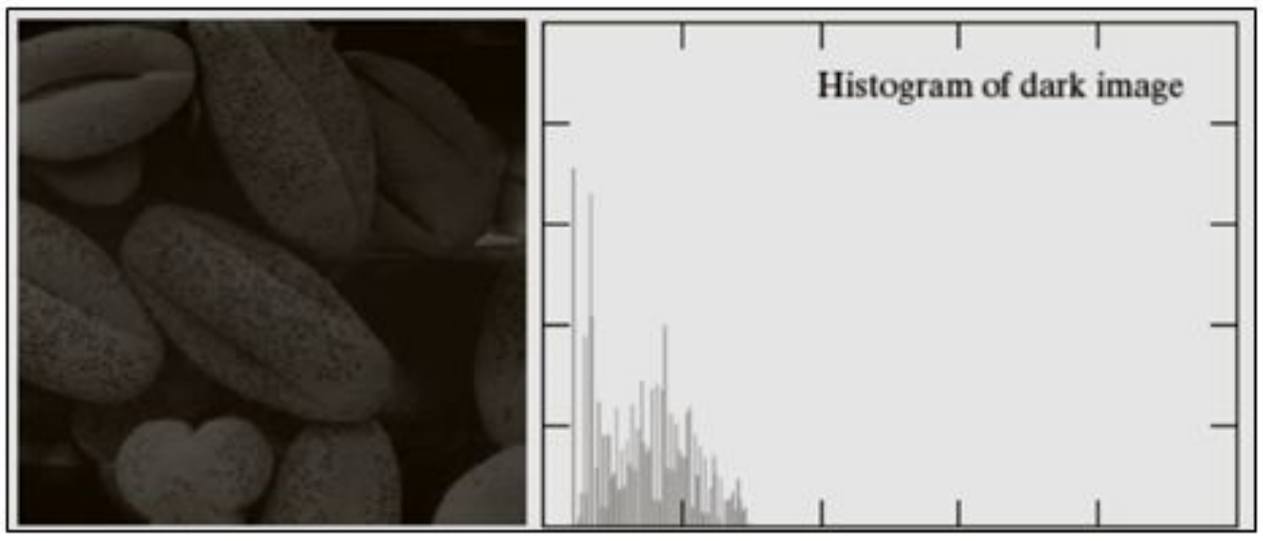
- **Global Transformation**

  全局变换

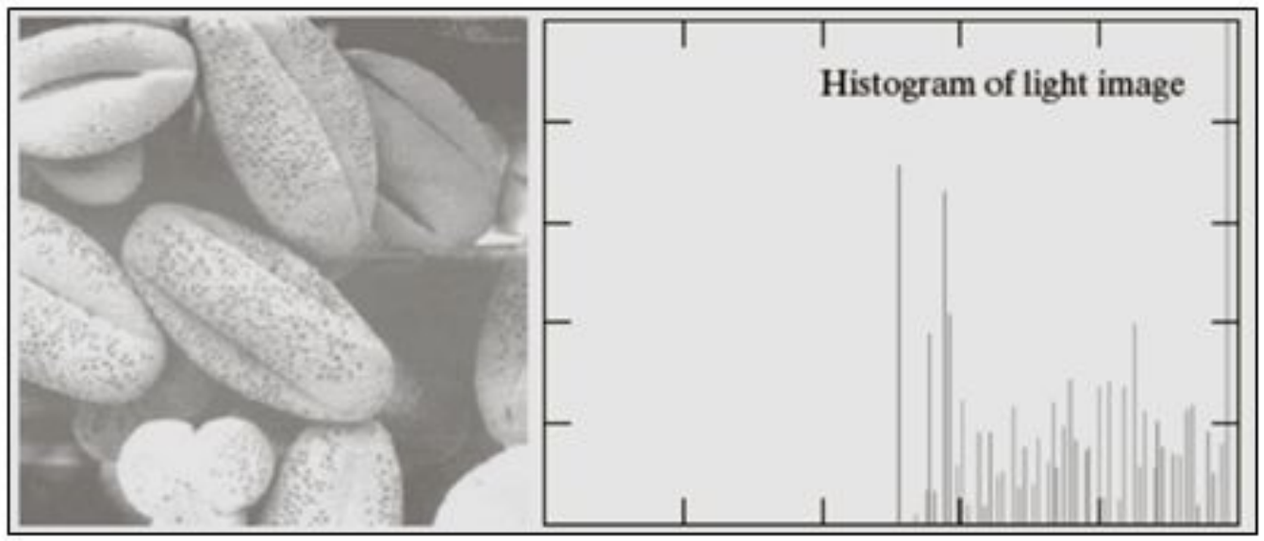
- **Local Transformation**

  局部变换

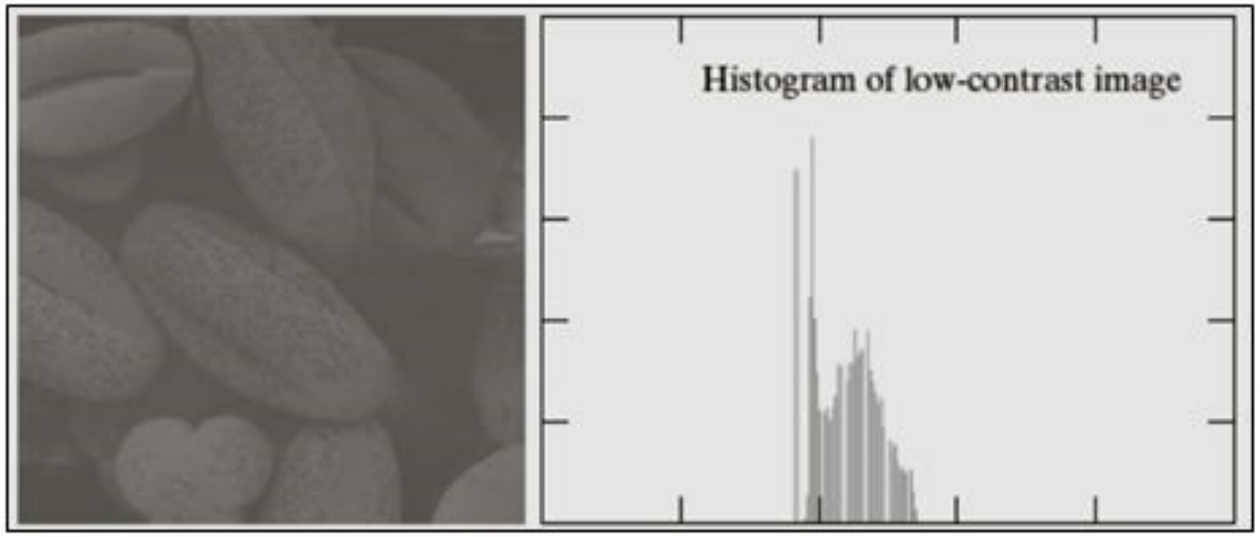
Image Histograms



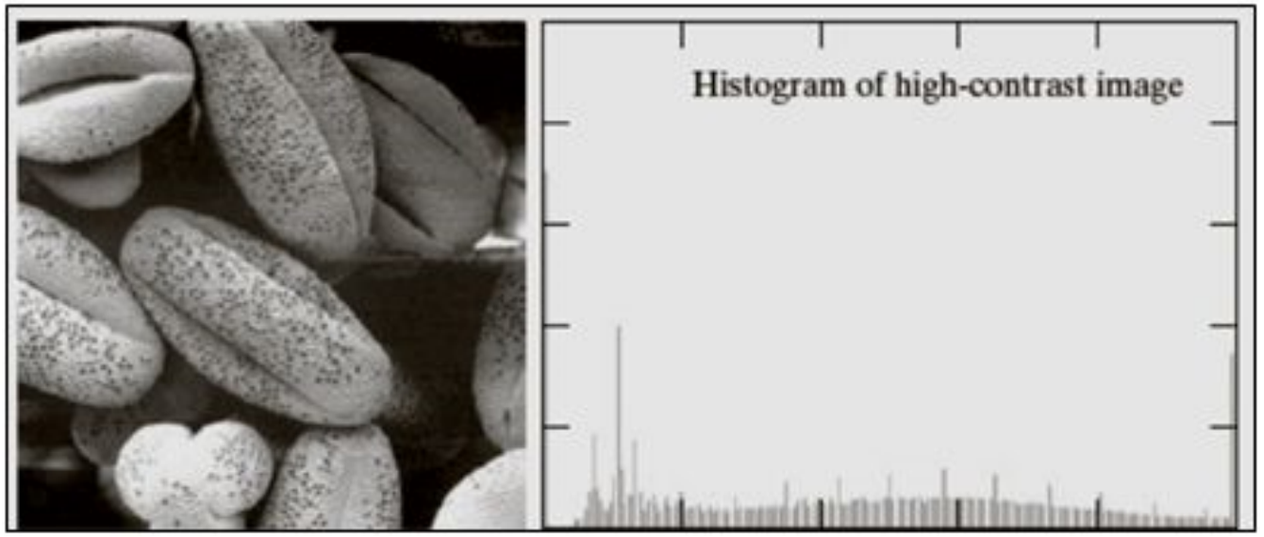
Dark



Light



Low-contrast



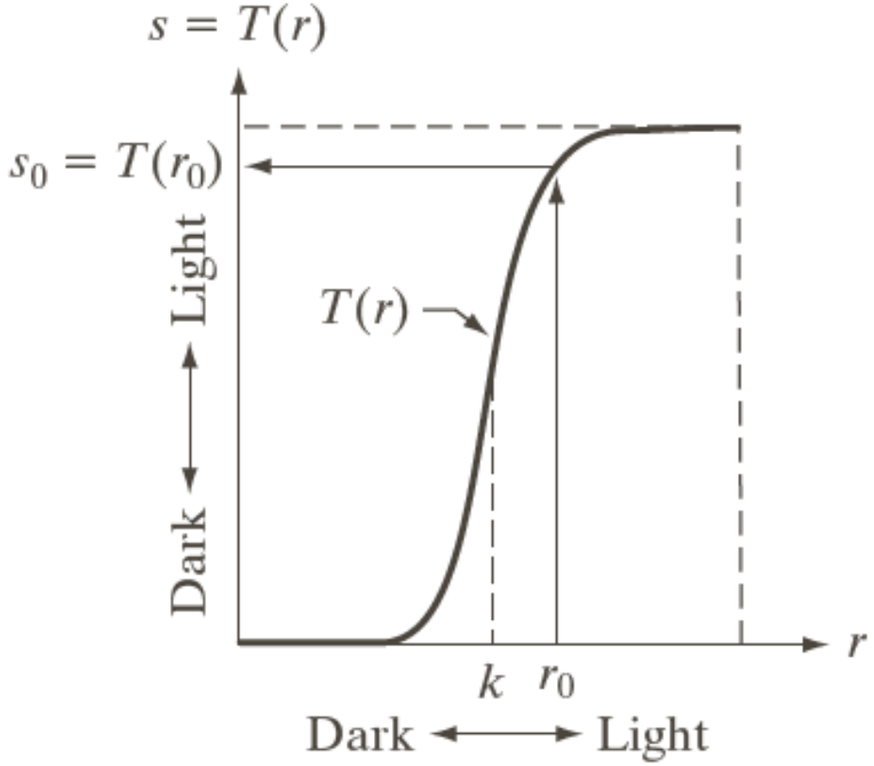
High-contrast

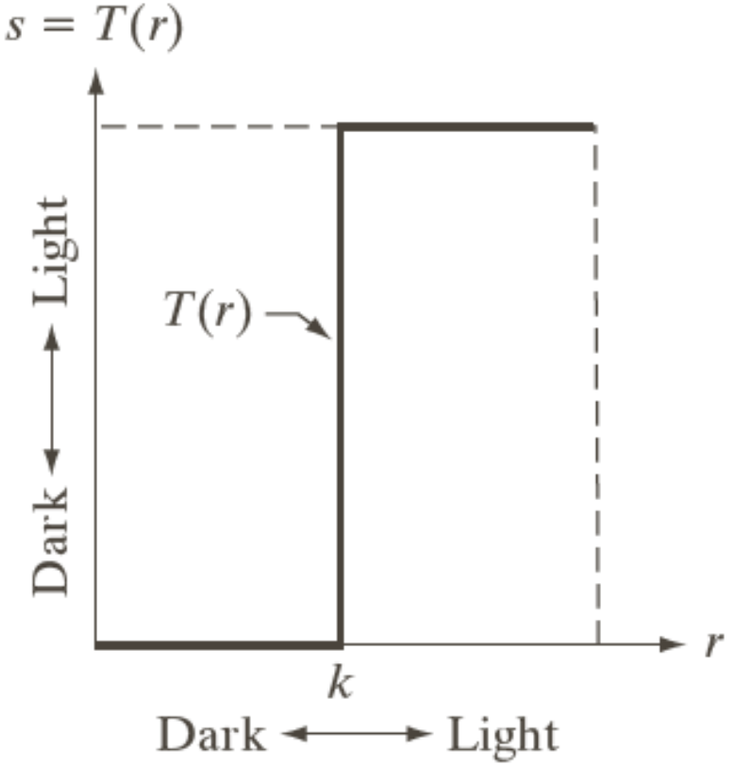
Spatial Domain Process

Global Transform

- **Intensity Transformation Function**

$$s = T(r)$$

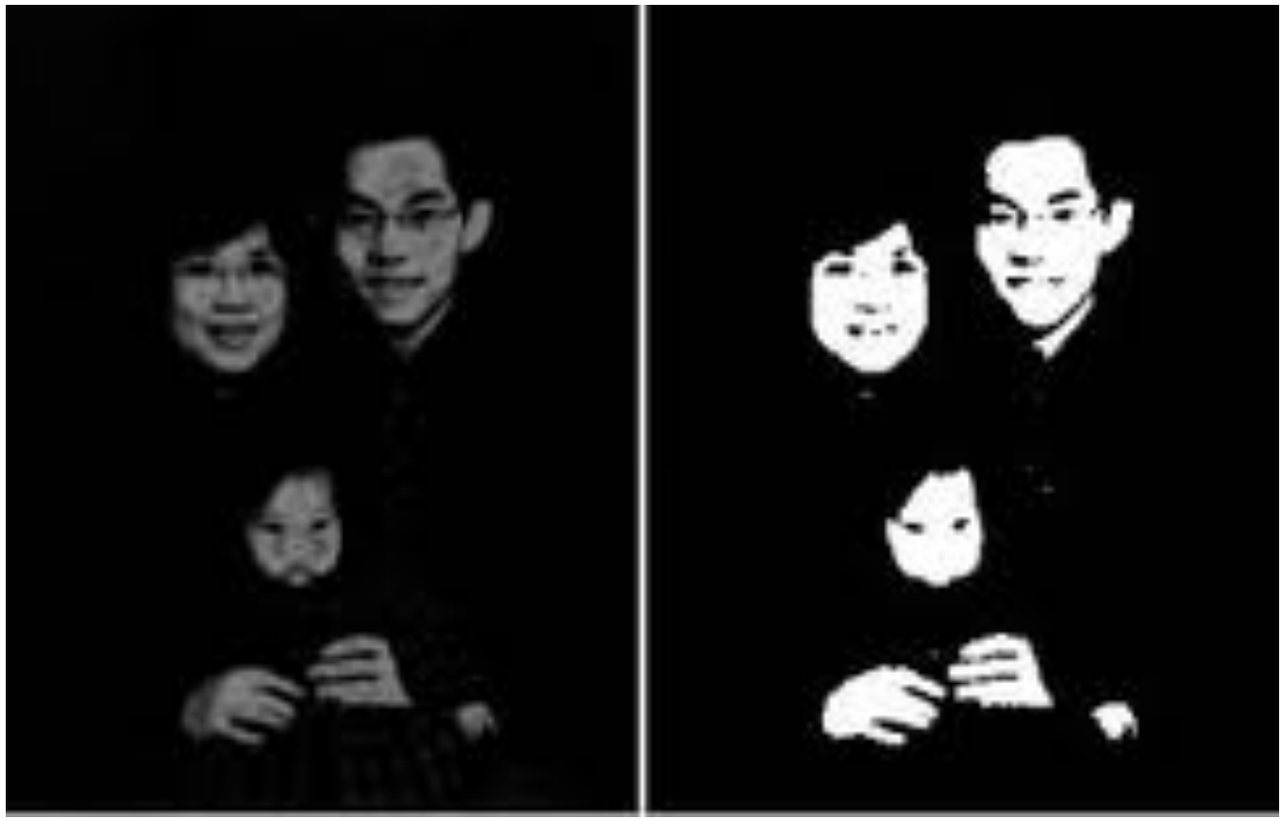


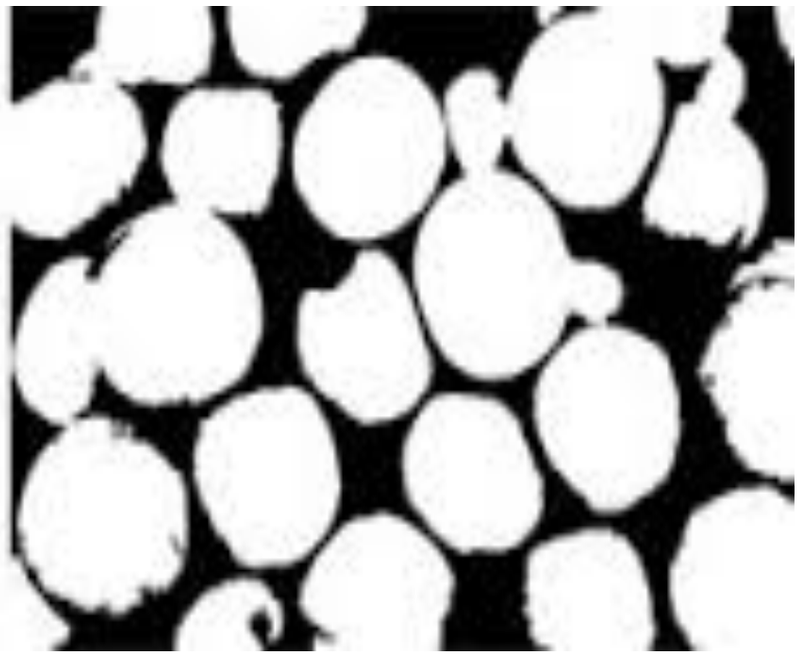


a b

FIGURE 3.2 Intensity transformation functions. (a) Contrast-stretching function. (b) Thresholding function.

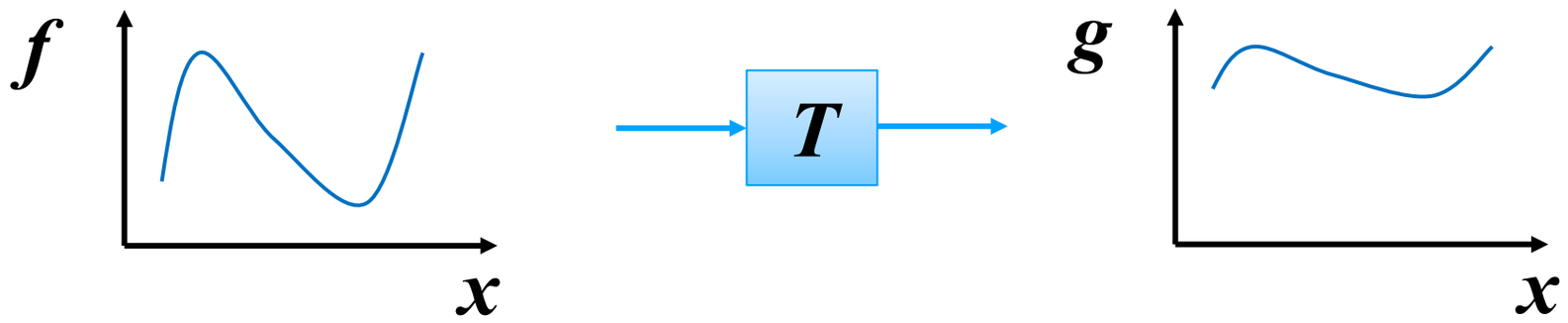
Examples



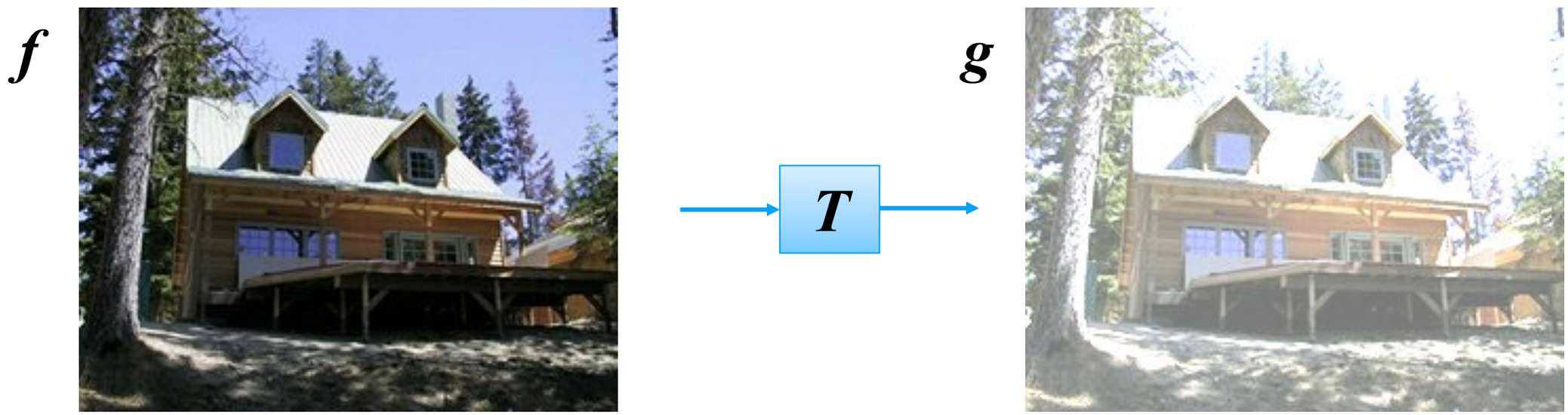


Range Transformations

- Change **range** of the signal $g(x) = T(f(x))$



- Change **range** of the image $g(x) = T(f(x))$



Basic Point Operator

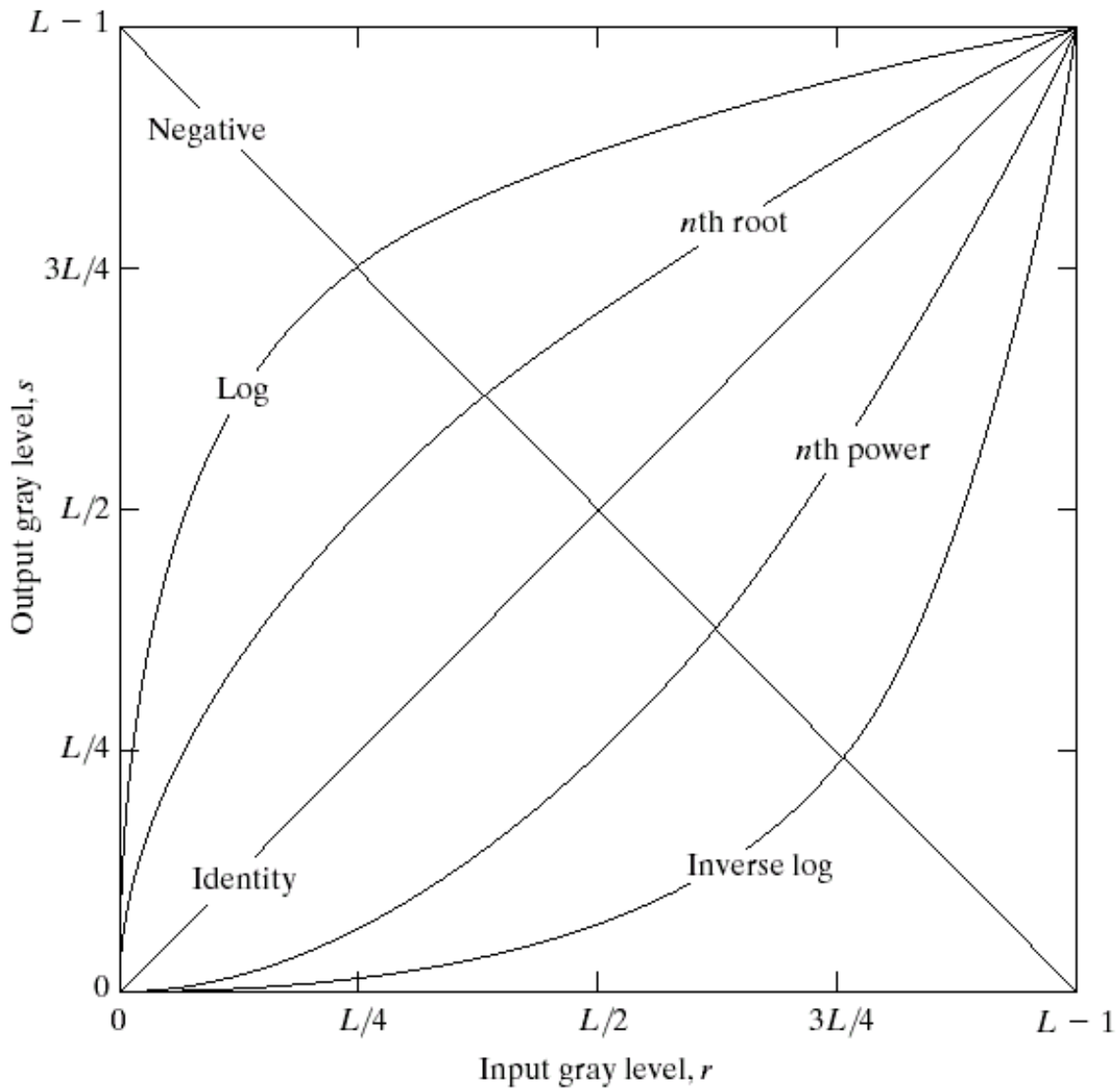
- **Basic Gray-Level Transformation Functions**

→ Image Enhancement

- Gamma Correction

- Negative Image

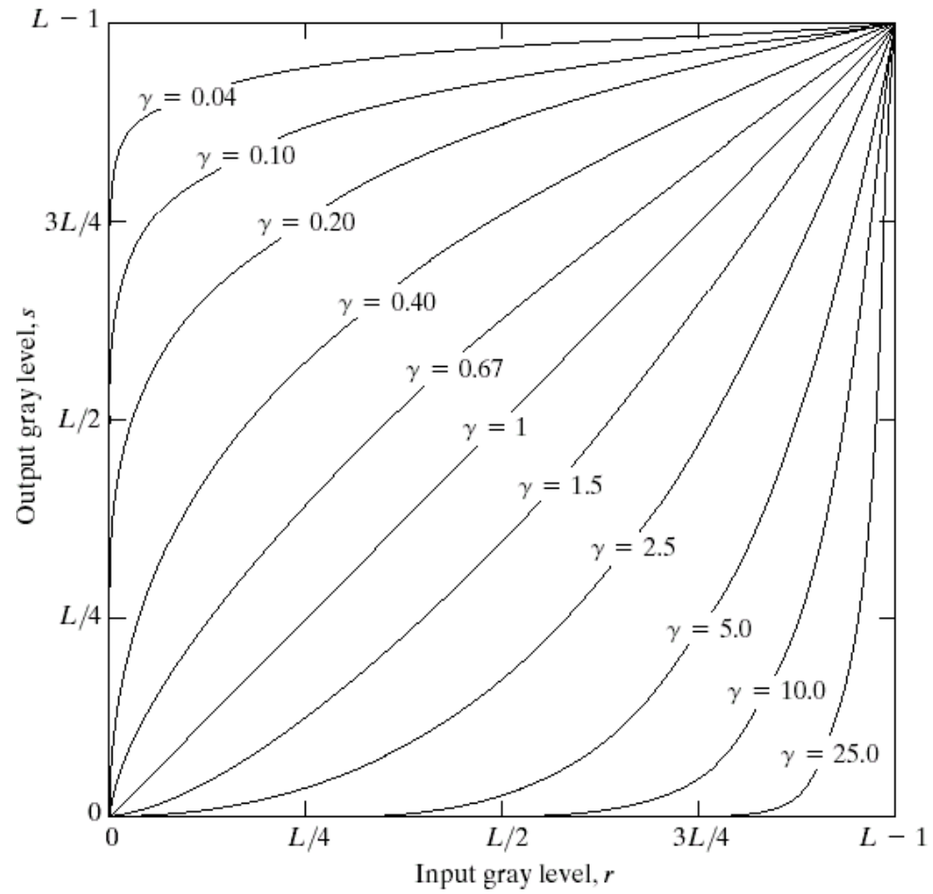
- Log Transform



Gamma Correction

- **Gamma Correction** is a family

$$s = c \cdot r^{\gamma}$$



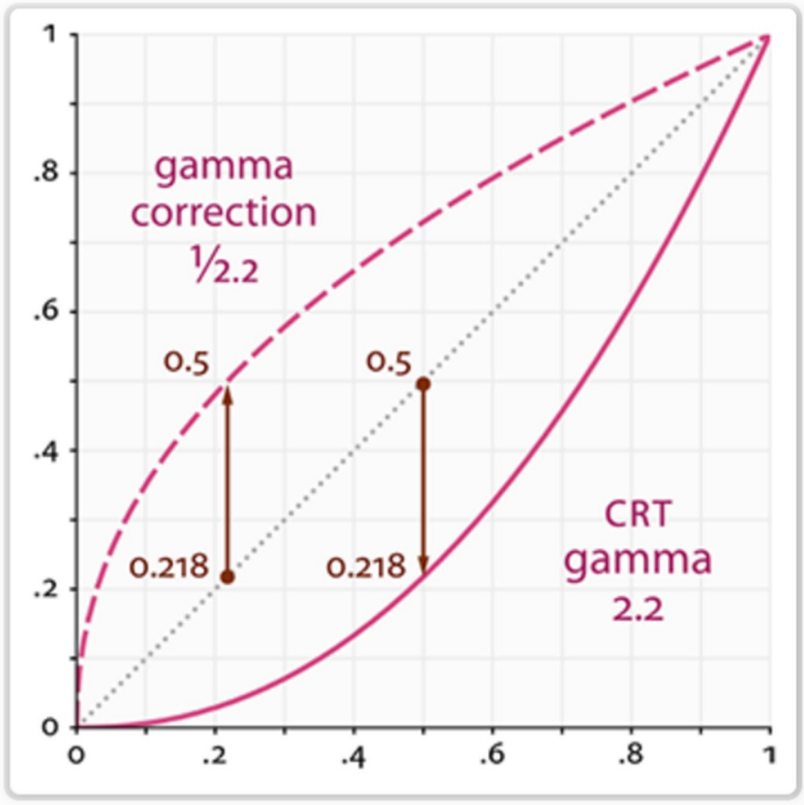
Gamma Correction

- **Gamma Correction** is a family

$$s = c \cdot r^{\wedge}\gamma$$

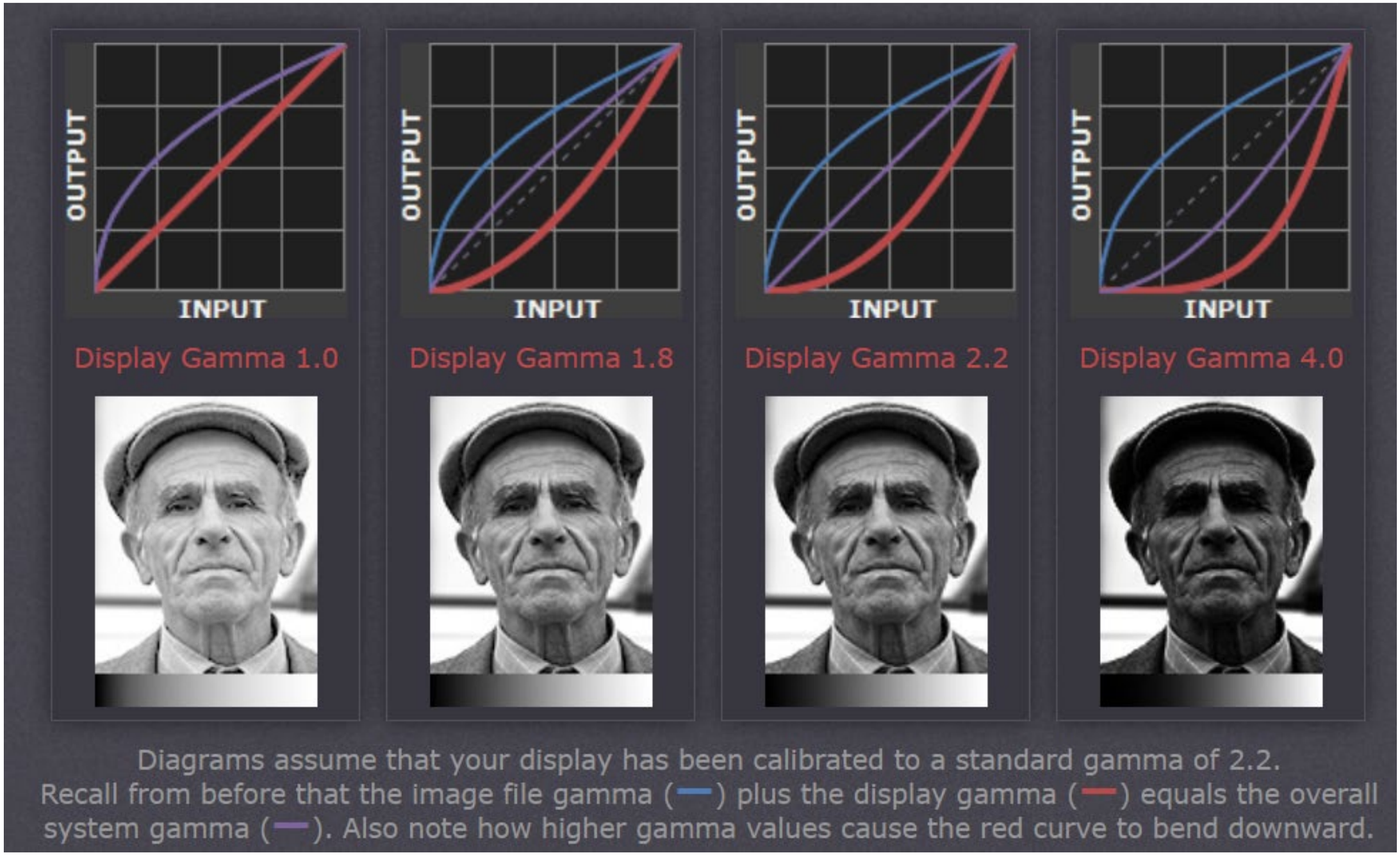
- Gamma = 2.2 for a display device

- Gamma = 1/2.2 for encoding (perceptually uniform encoding)



Why Gamma Is Useful

- **Our eyes do not perceive light the way cameras do**

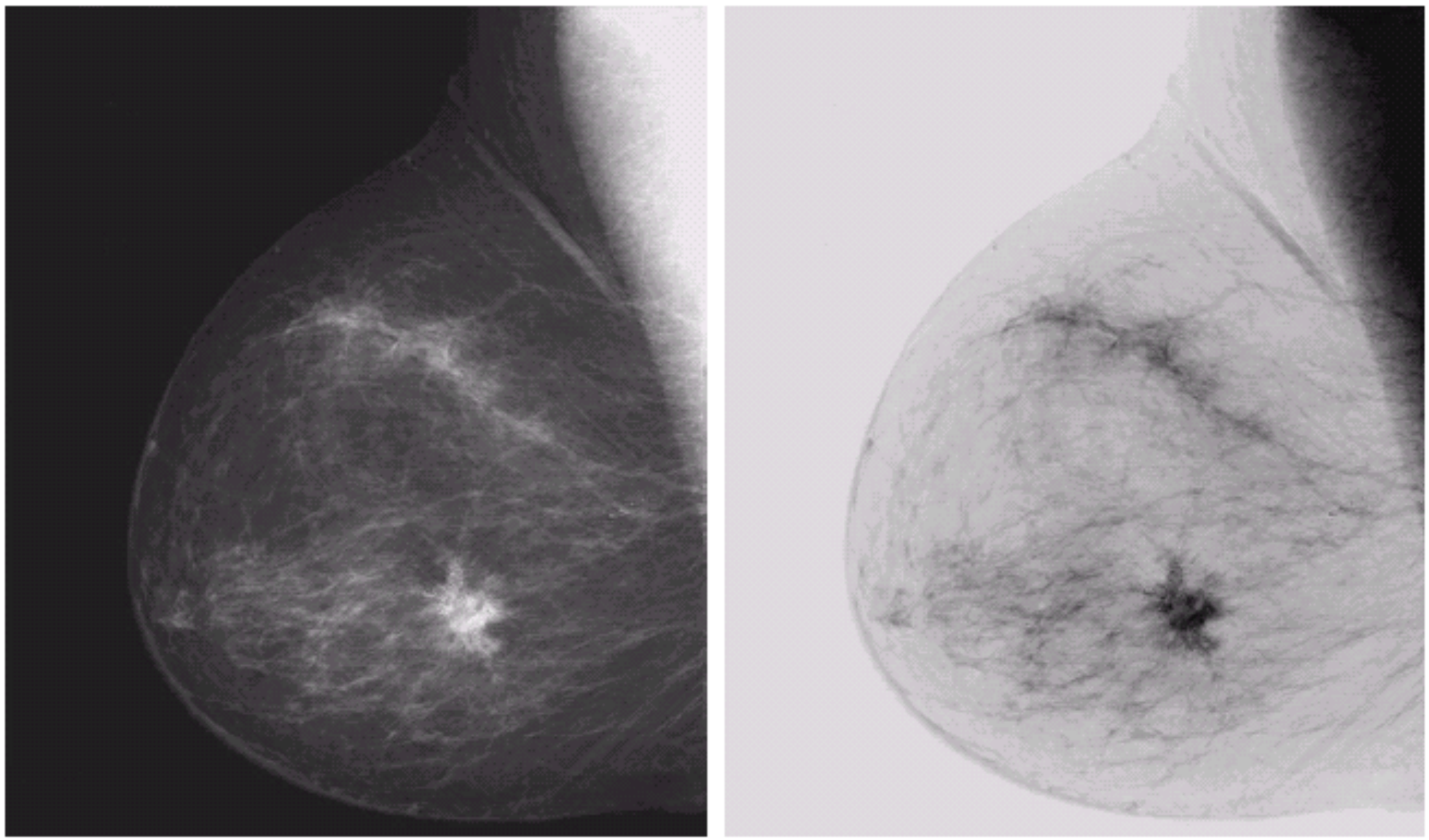


Source: https://www.cambridgeincolour.com/tutorials/gamma-correction.htm

Negative Transformation

- **Image Negatives**

$$s = L - 1 - r$$



a b

FIGURE 3.4 (a) Original digital mammogram. (b) Negative image obtained using the negative transformation in Eq. (3.2-1). (Courtesy of G.E. Medical Systems.)

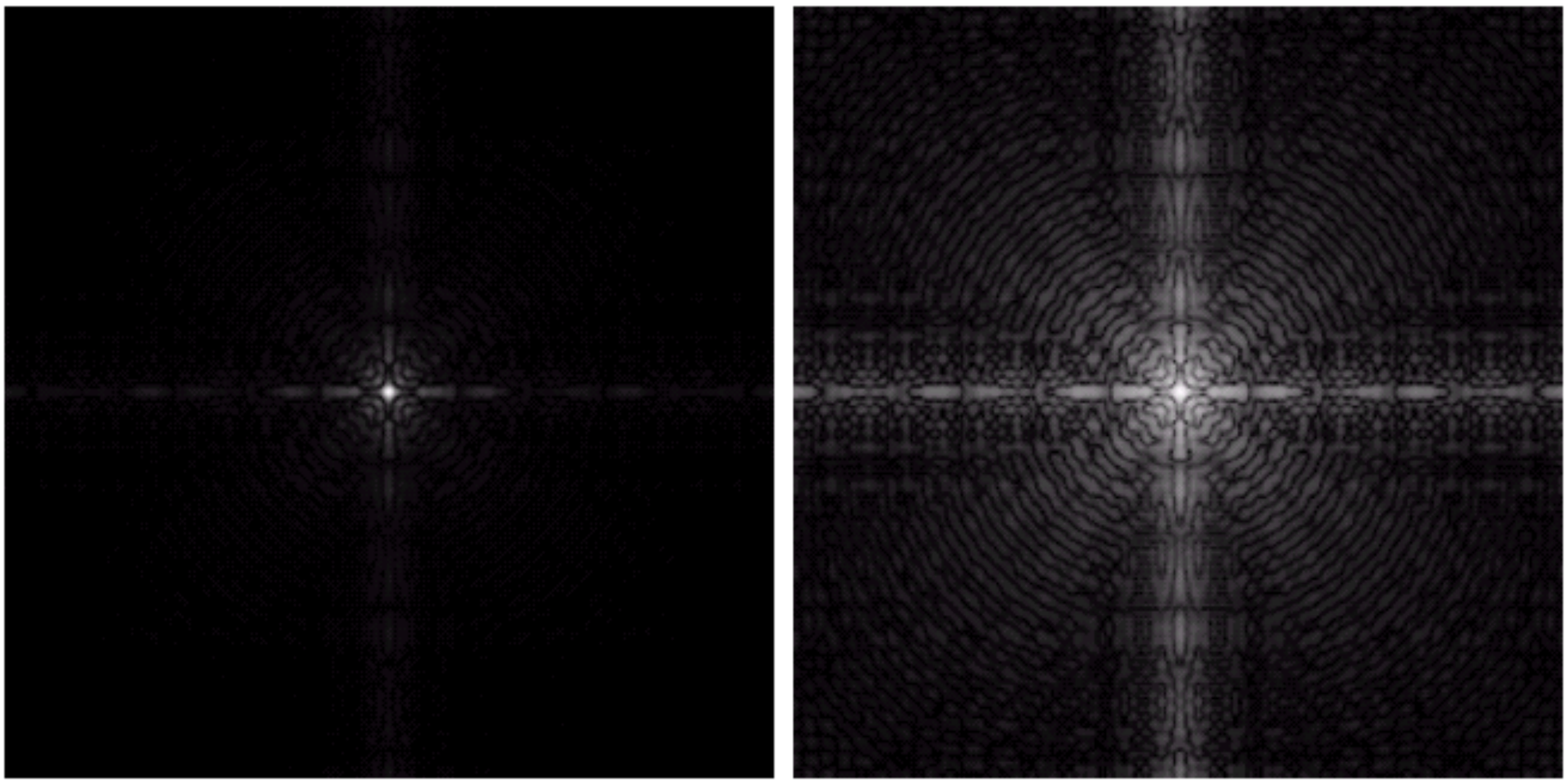
Log Transformation

- **Log Transformations**

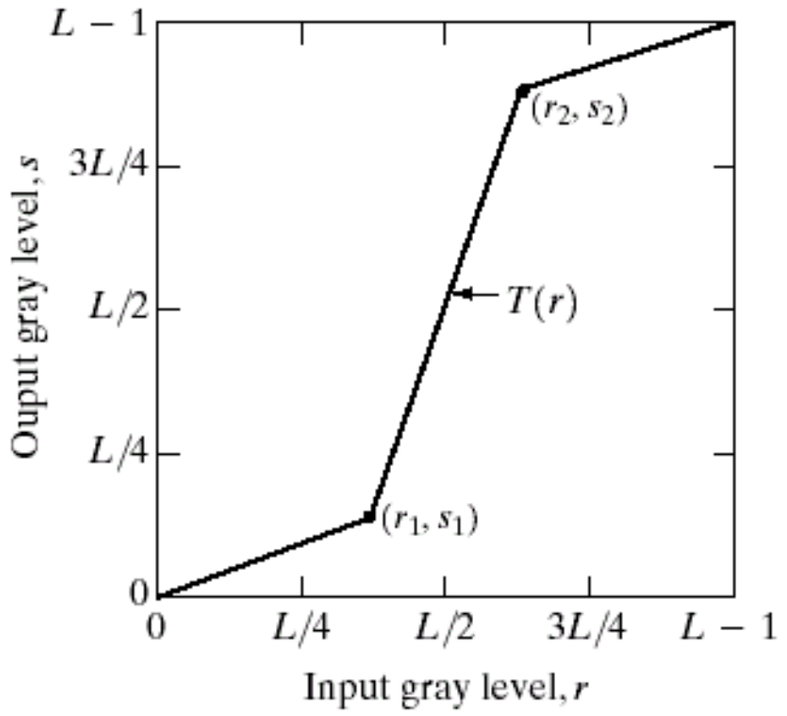
$$s = c\log(1+r)$$

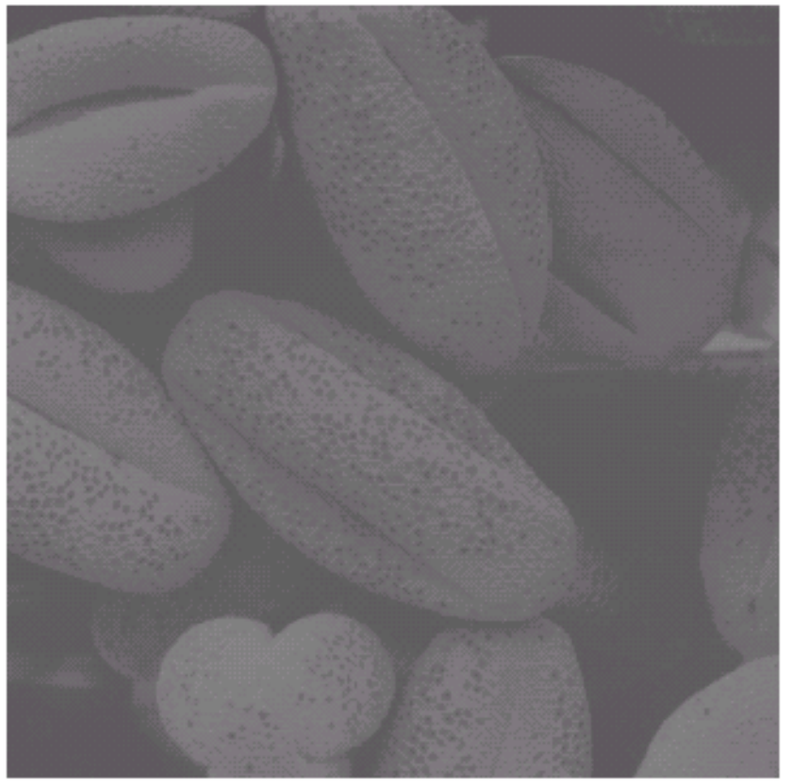
a b

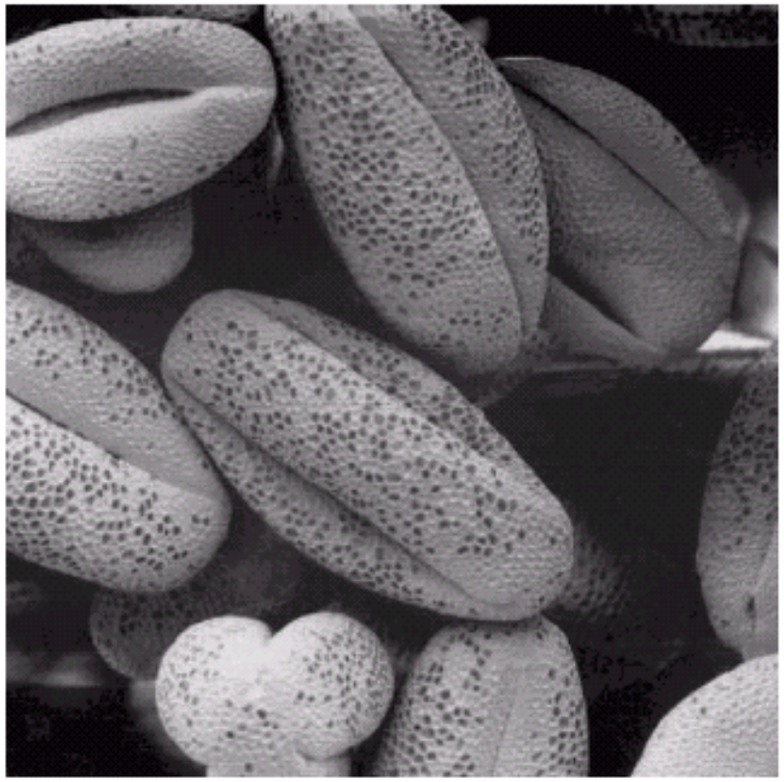
FIGURE 3.5
(a) Fourier spectrum.
(b) Result of applying the log transformation given in Eq. (3.2-2) with $c = 1$.



‘Contrast’ Stretching





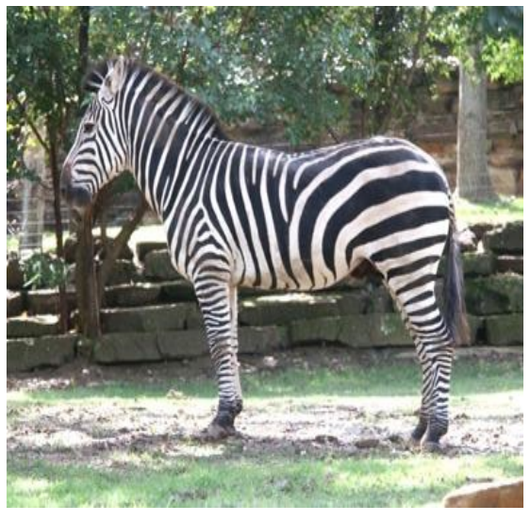
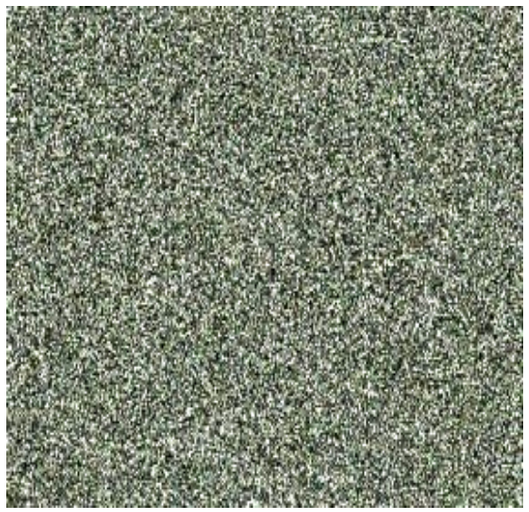


a b
c d

FIGURE 3.10
Contrast stretching. (a) Form of transformation function. (b) A low-contrast image. (c) Result of contrast stretching. (d) Result of thresholding. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)

Image Neighborhoods

- What happens if we reshuffle all pixels within the images?

→ Its histogram won’t change.
Point-wise processing unaffected.

- Measure properties relative to small neighborhoods of pixels

Spatial Domain Process

Local Transform

- **Point & Neighborhood**

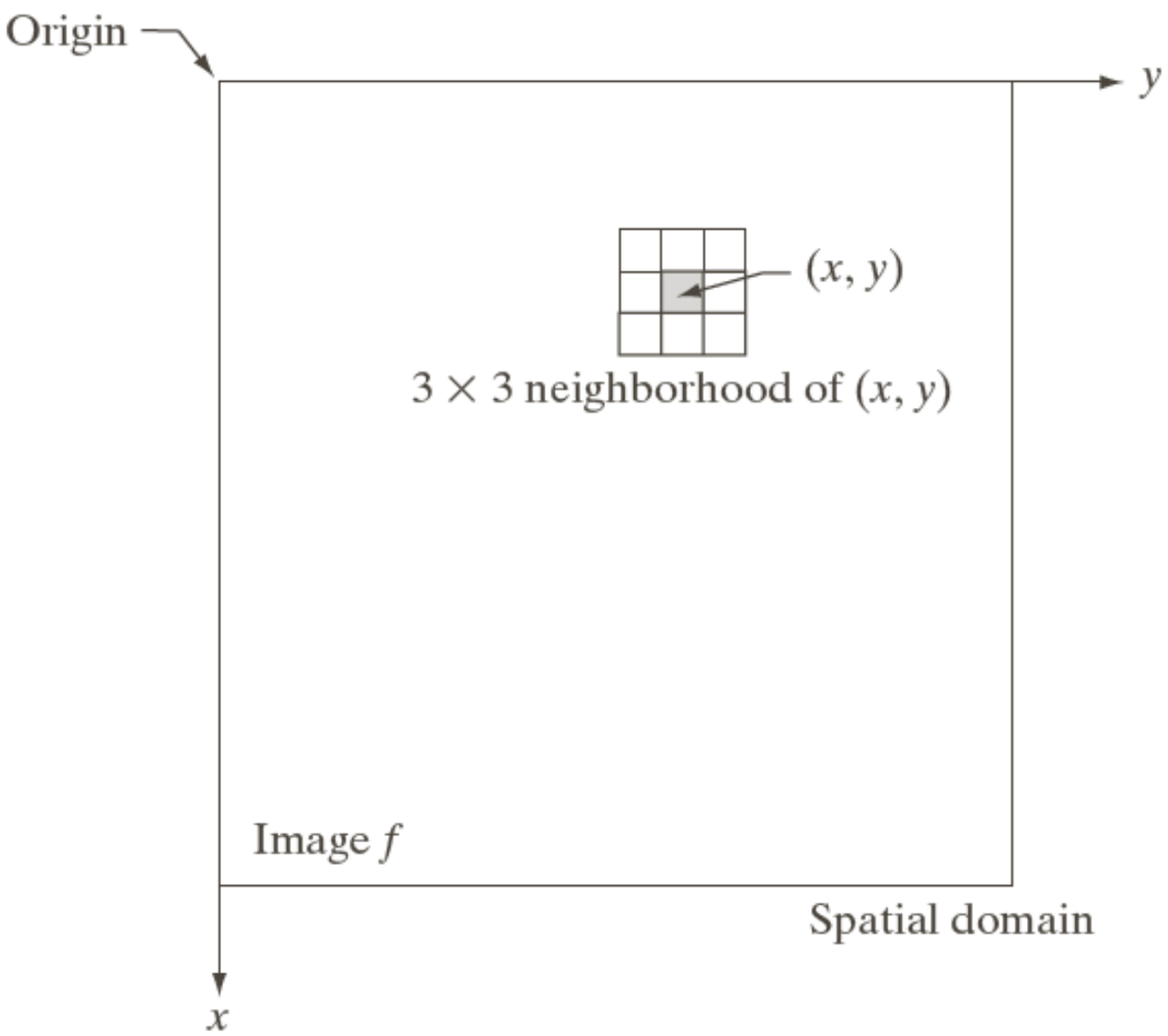


FIGURE 3.1
A 3×3 neighborhood about a point (x, y) in an image in the spatial domain. The neighborhood is moved from pixel to pixel in the image to generate an output image.

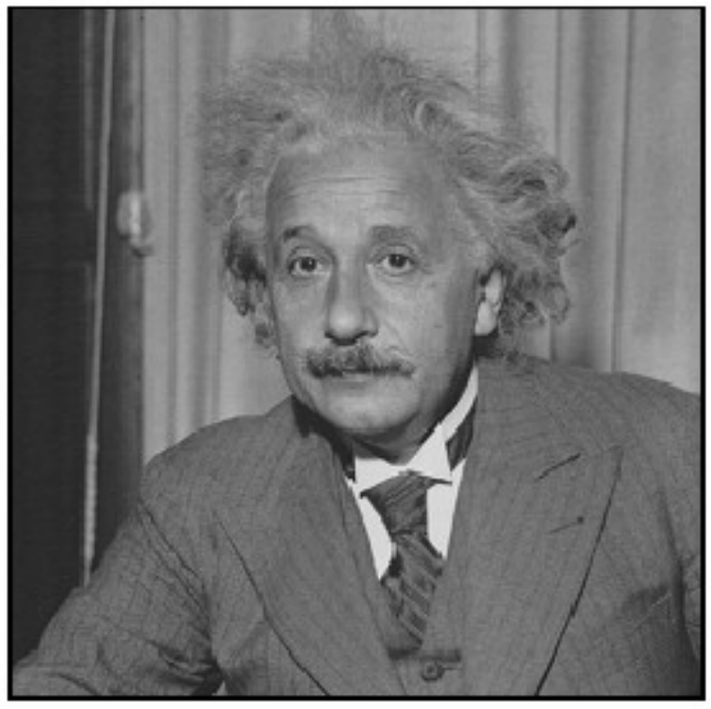
$$g(x, y) = T[f(x, y)])$$

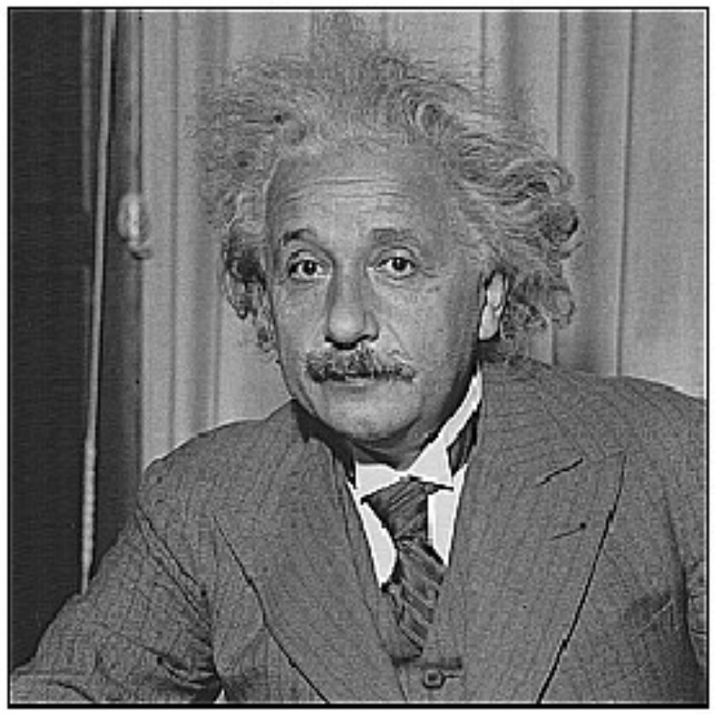
$f(x, y)$: input image

$g(x, y)$: output image

T : an operator on f defined over a neighborhood of point (x, y)

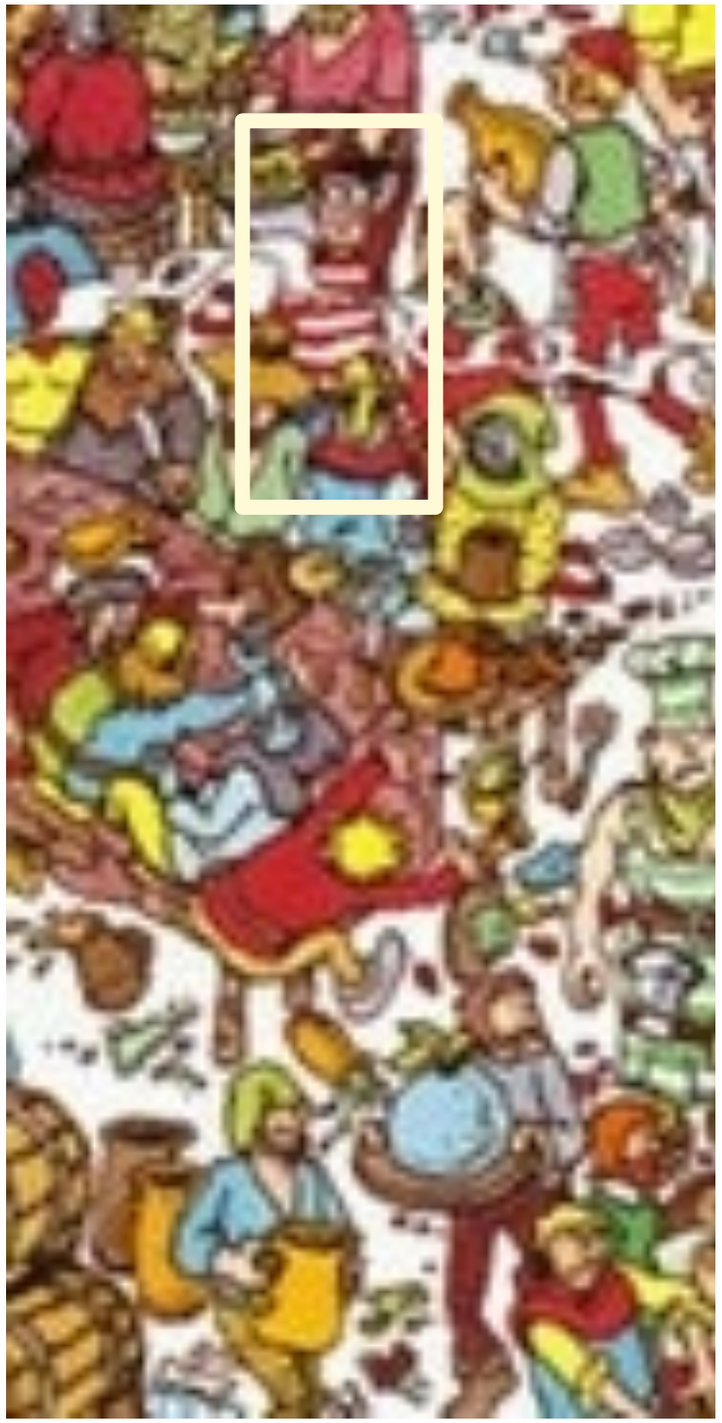
Image Filters





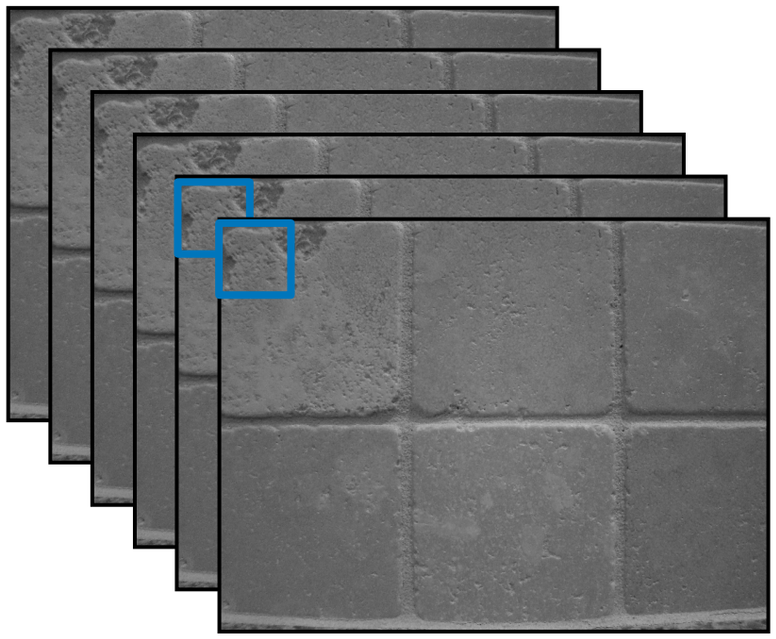
Smooth/Sharpen Images...

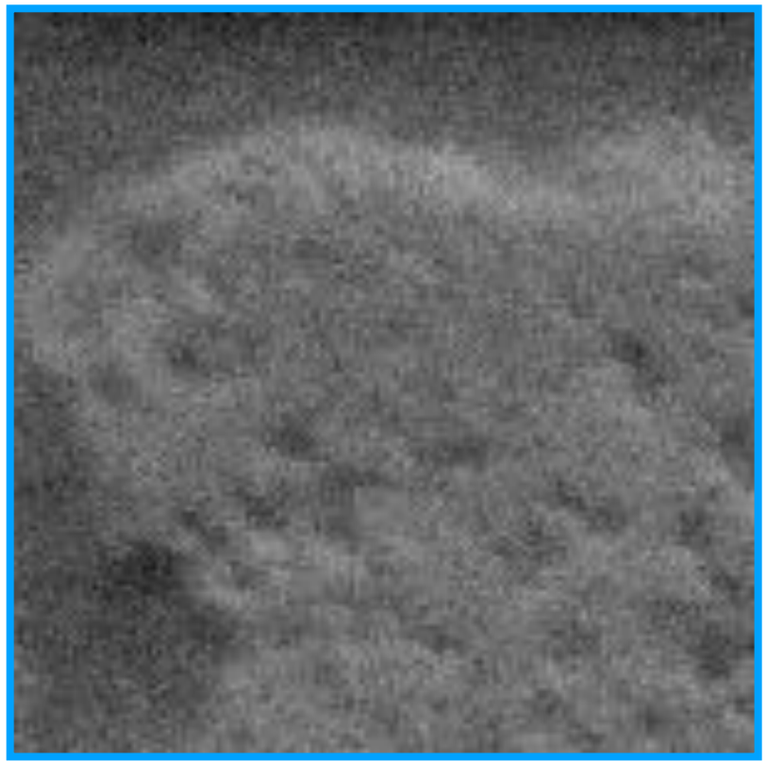
Find Edges...

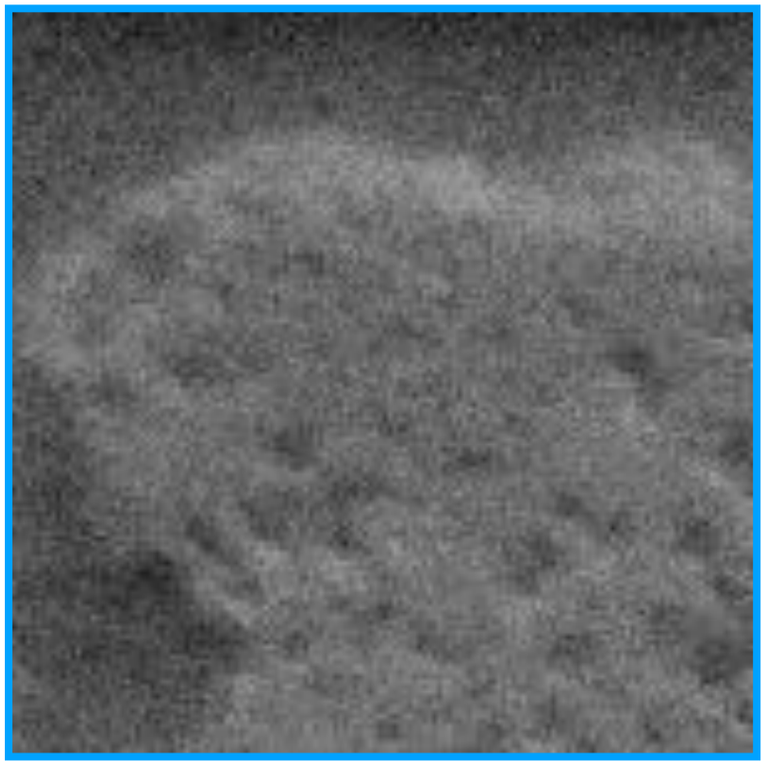


Find Waldo…

Motivation: Noise Reduction







- How could we reduce the noise,

→ Give an estimate of the true intensities?

- What if there’s only one image?

First Attempt at a Solution

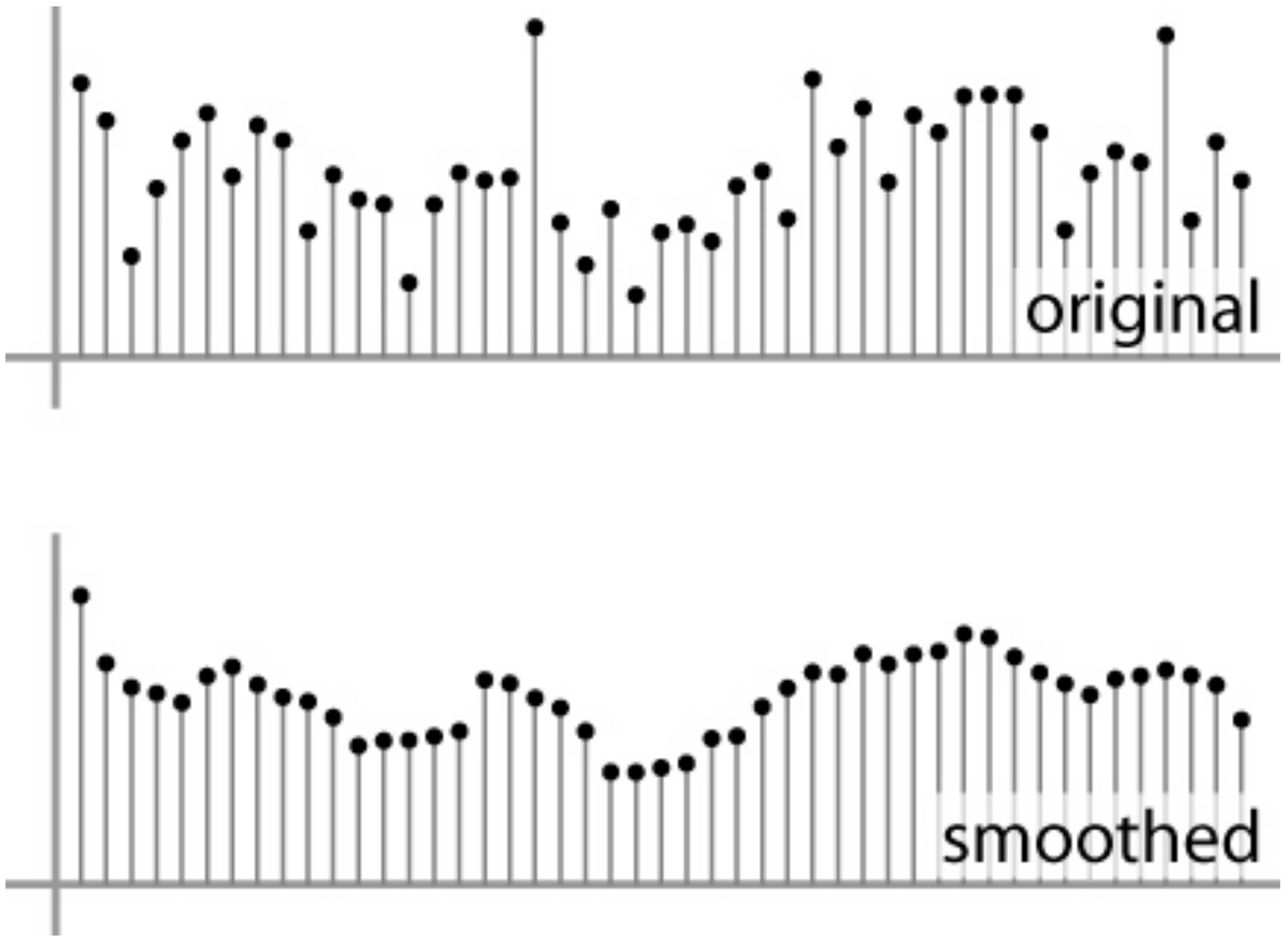
- Let's **replace each pixel** with an **average of all the values in its neighborhood**
- **Assumptions:**

  - Expect pixels to be like their neighbors

  - Expect noise processes to be independent from pixel to pixel

First Attempt at a Solution

- Let’s replace each pixel with an average of all the values in its neighborhood
- Moving average in 1D:

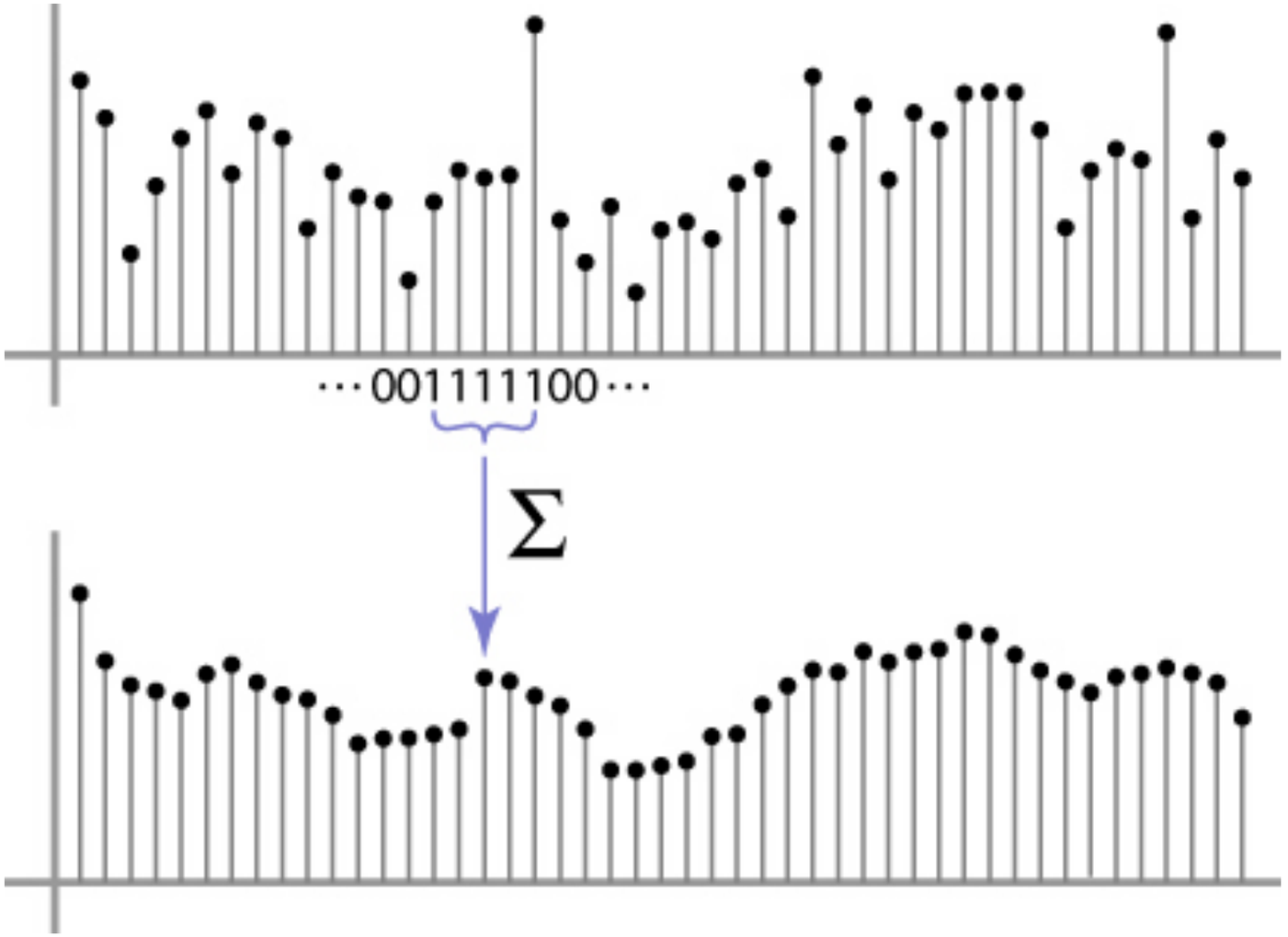


Source: S. Marschner

Weighted Moving Average

- Can add weights to our moving average

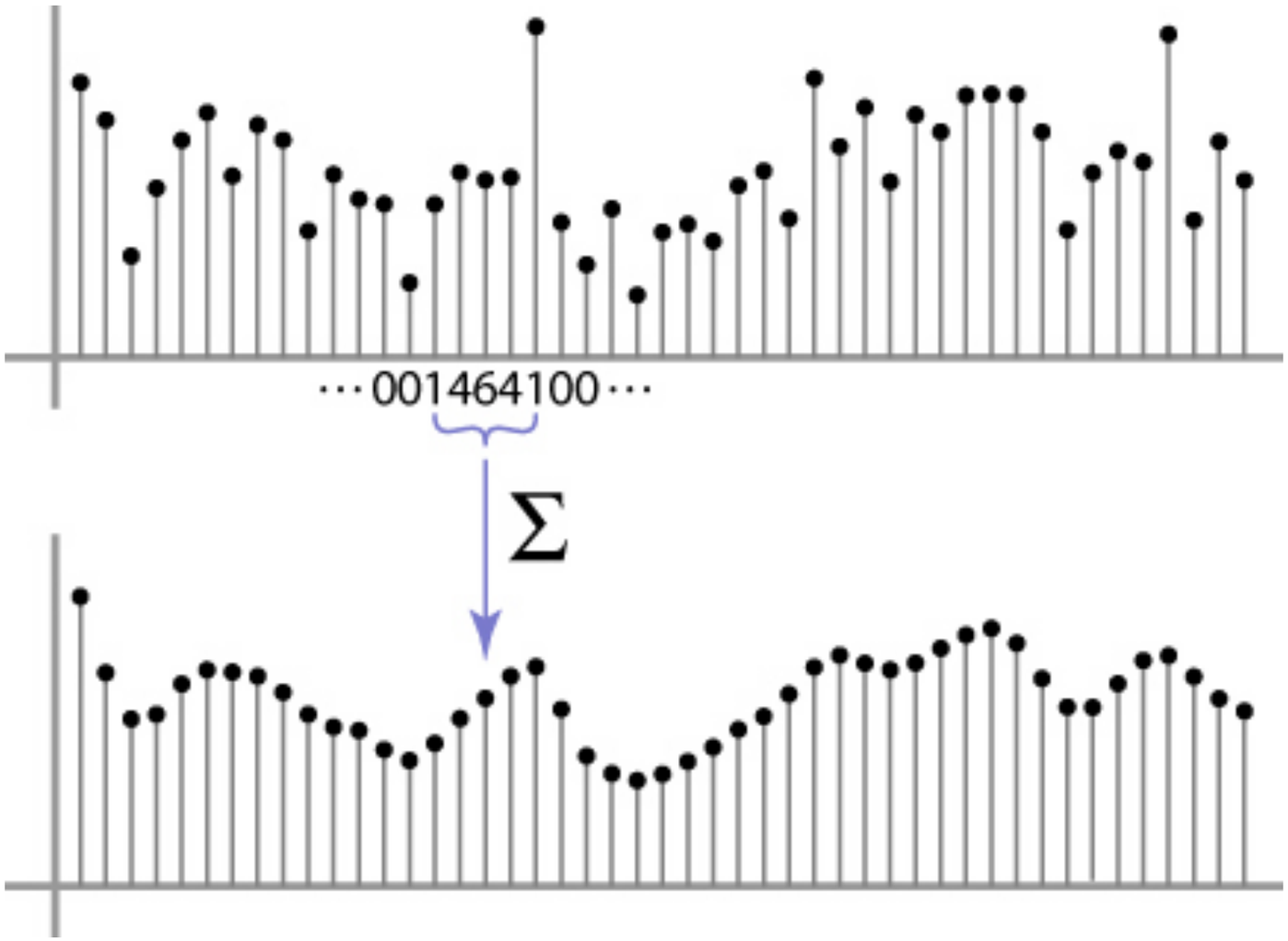
  Weights [1, 1, 1, 1, 1] / 5



Source: S. Marschner

Weighted Moving Average

- Non-uniform Weights **[1, 4, 6, 4, 1] / 16**



Source: S. Marschner

Moving Average In 2D

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0								

Source: S. Seitz

Moving Average In 2D

$$F[x, y]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$G[x, y]$$

	0	10							

Source: S. Seitz

Moving Average In 2D

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20						

Source: S. Seitz

Moving Average In 2D

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30					

Source: S. Seitz

Moving Average In 2D

$$F[x, y]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$G[x, y]$$

	0	10	20	30	30				

Source: S. Seitz

Moving Average In 2D

$$F[x, y]$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$G[x, y]$$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Source: S. Seitz

Correlation Filtering

- Say the averaging window size is (2k+1) x (2k+1):

$$G[i,j] = \underbrace{\frac{1}{(2k+1)^2}}_{\text{Attribute uniform weight to each pixel}} \underbrace{\sum_{u=-k}^{k}\sum_{v=-k}^{k} F[i+u, j+v]}_{\text{Loop over all pixels in neighborhood image around pixel } F[i,j]}$$

- Now generalize to allow **different weights** depending on neighboring pixel's relative position:

$$G[i,j] = \sum_{u=-k}^{k}\sum_{v=-k}^{k} \underbrace{H[u,v]}_{\text{Non-uniform weights}} F[i+u, j+v]$$

Correlation Filtering

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$$

- This is called **cross-correlation** (互相关), denoted

$$G = H \otimes F$$

- Filtering an image: Replace each pixel with a linear combination of its neighbors.
- The filter **kernel** / **mask** $H[u, v]$ is the prescription for the weights in the linear combination.

Averaging Filter

- What values belong in the kernel H for the moving average example?

$$G = H \otimes F$$

$F[x, y]$ $\otimes$ $H[u, v]$ $G[x, y]$

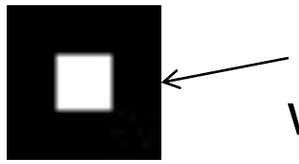
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

	0	10	20	30	30				

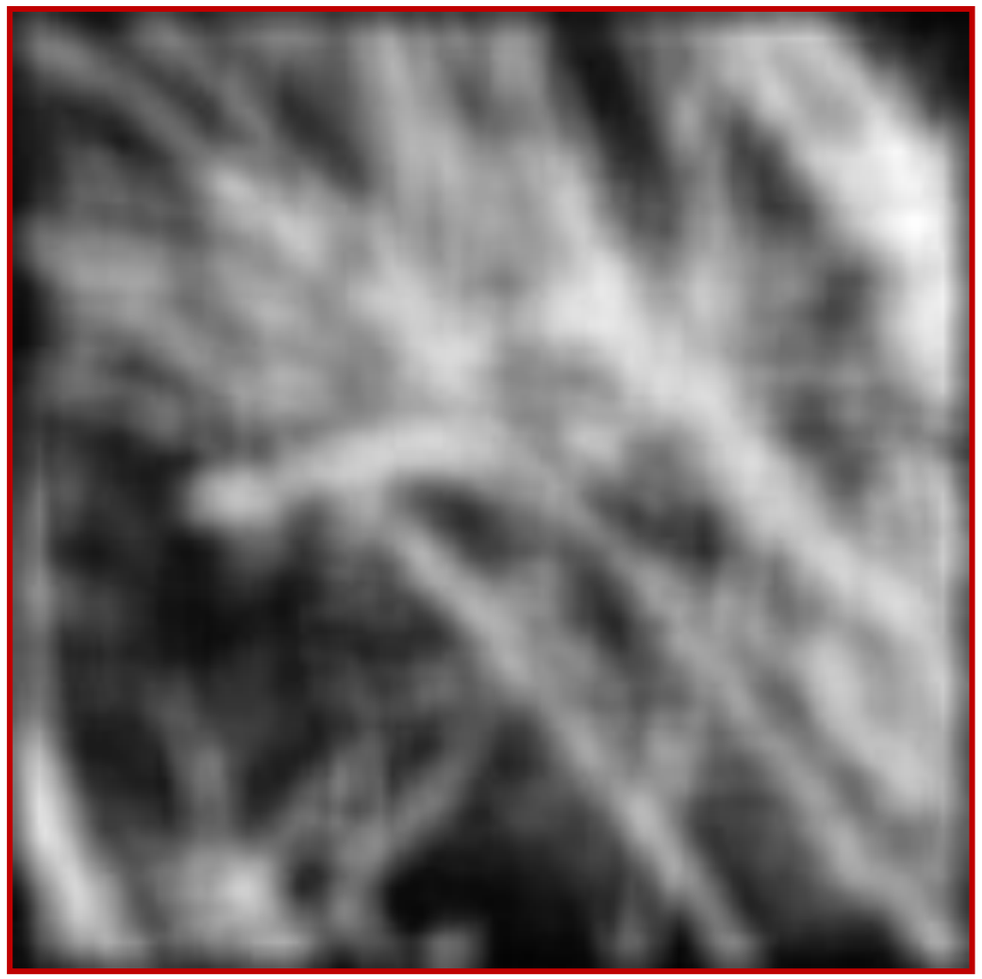
Smoothing by Averaging



Box Filter:

white = high value, black = low value

original



filtered

Gaussian Filter

- What if we want nearest neighboring pixels to have the most influence on the output?

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$F[x, y]$

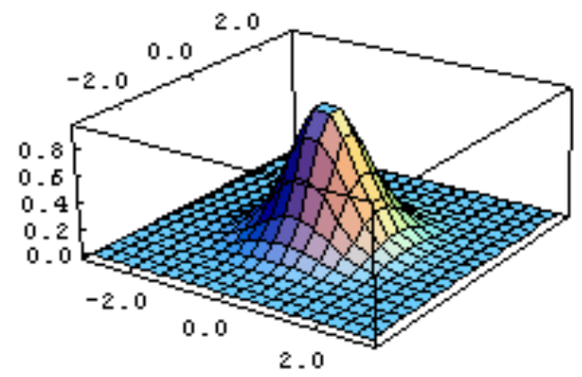
$$\frac{1}{16}$$

1	2	1
2	4	2
1	2	1

$H[u, v]$

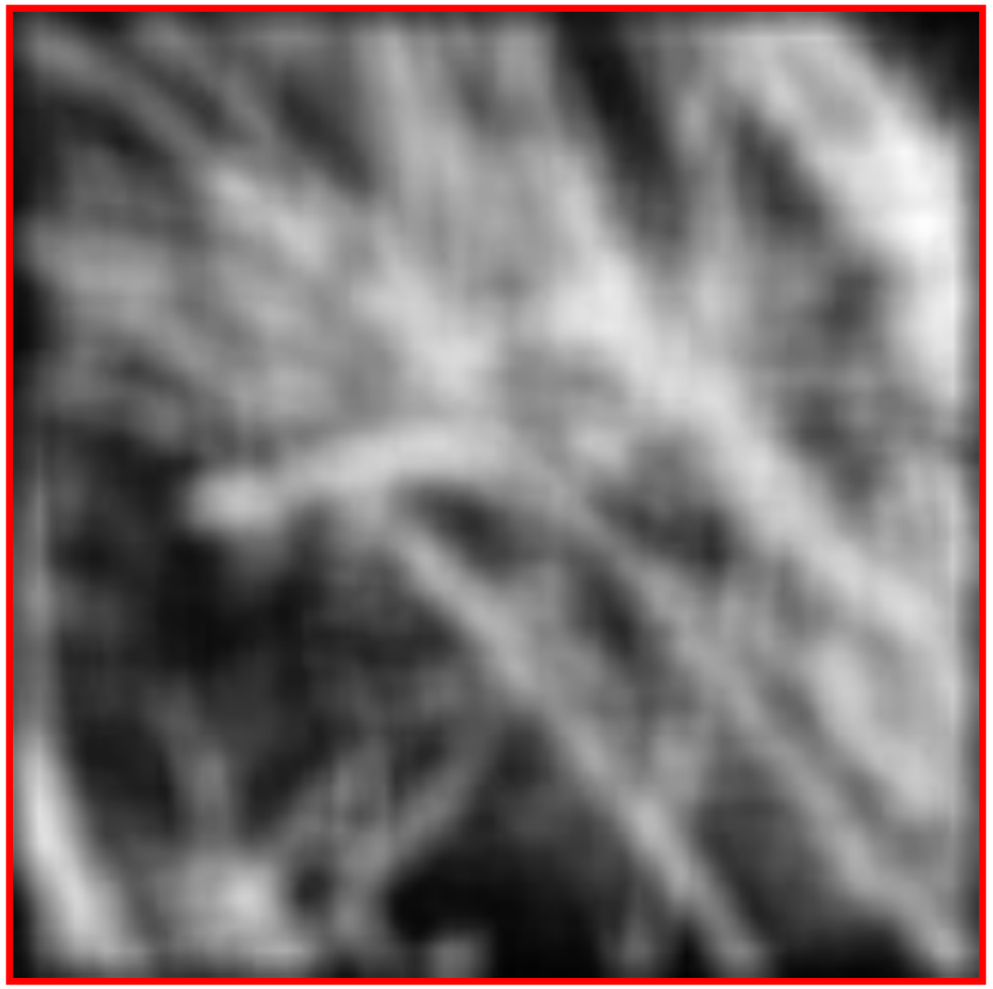
This kernel is an approximation of a Gaussian function:

$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



Source: S. Seitz

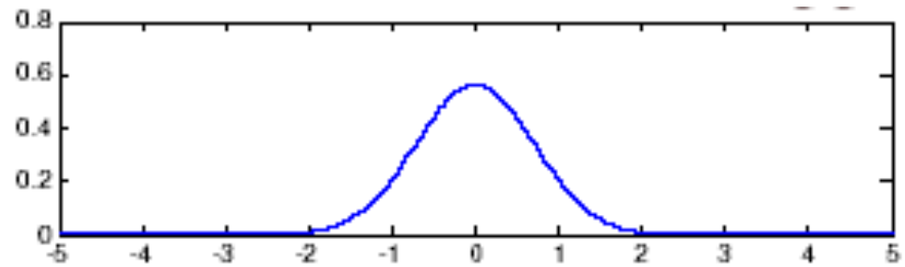
Smoothing with a Gaussian



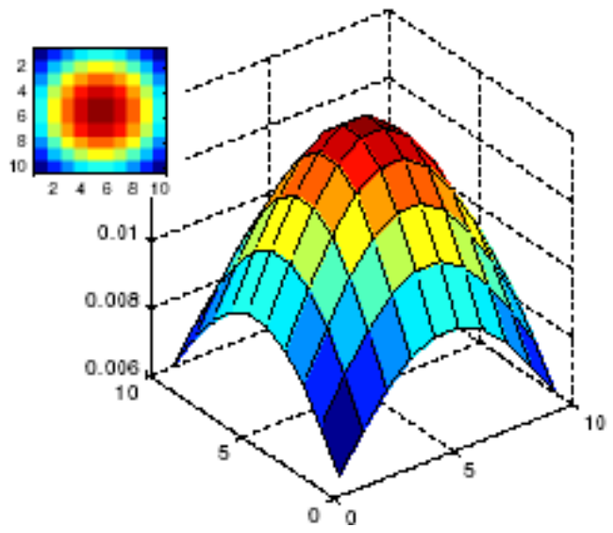
Gaussian Filters

- What parameters matter here?
- **Size** of kernel or mask

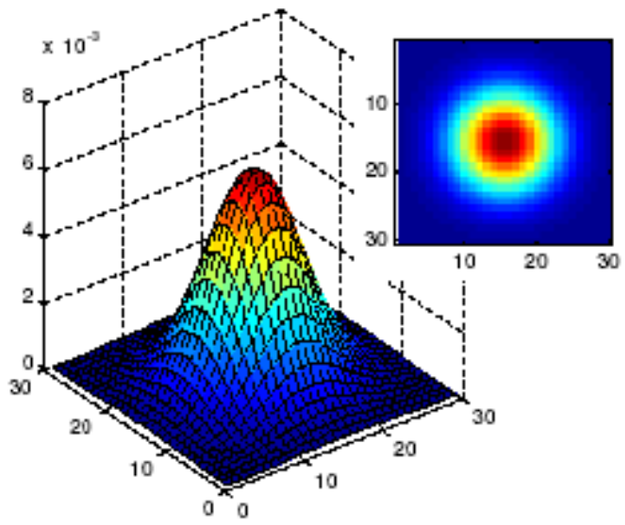
  - Note: Gaussian function has infinite support,



but discrete filters use finite kernels



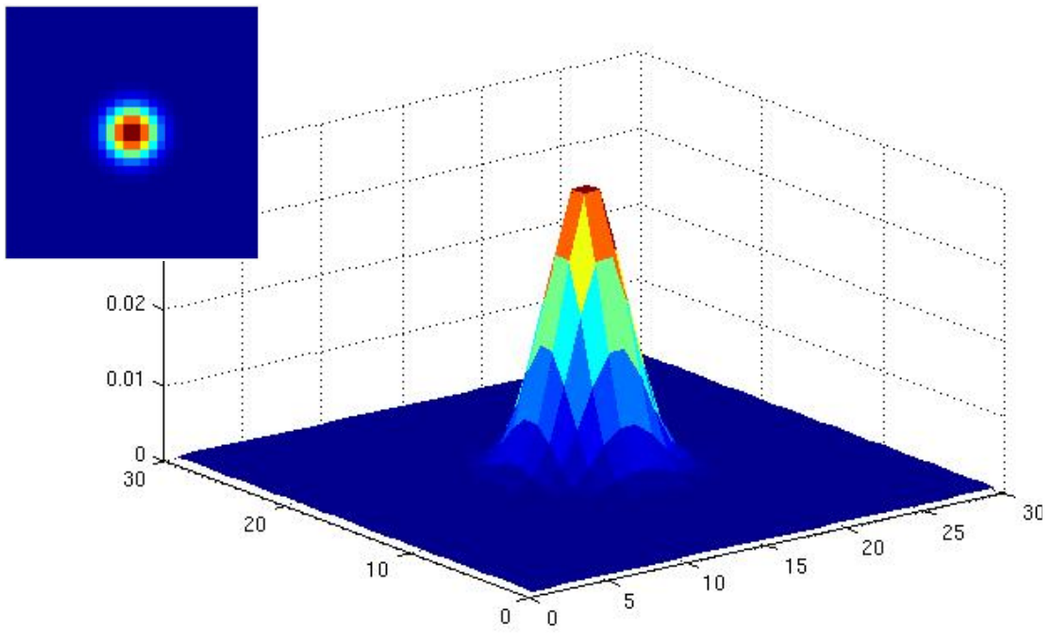
$\sigma = 5$ with 10×10 kernel



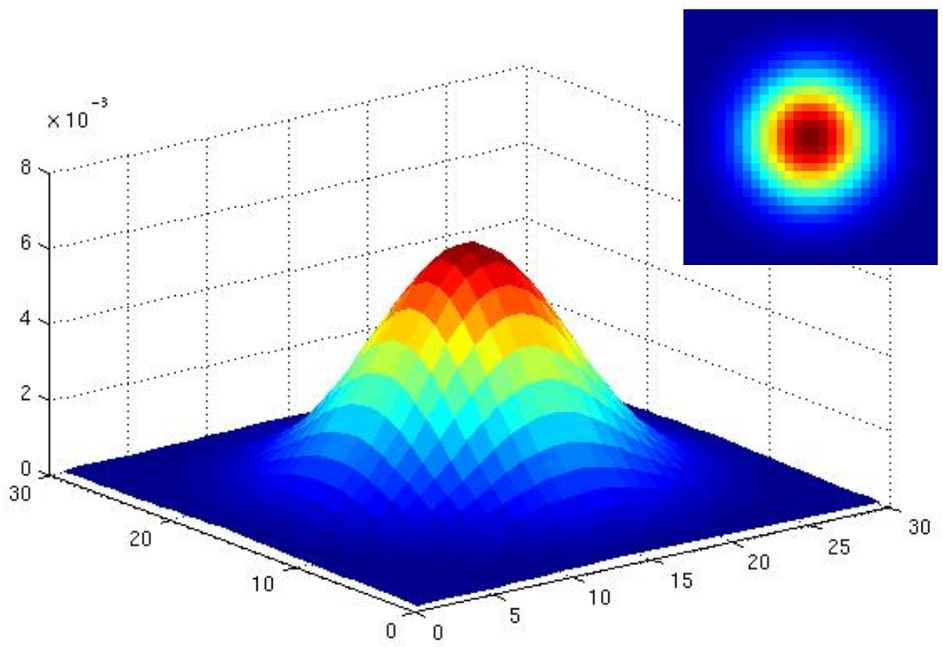
$\sigma = 5$ with 30×30 kernel

Gaussian Filters

- What parameters matter here?
- **Variance** of Gaussian: determines extent of smoothing

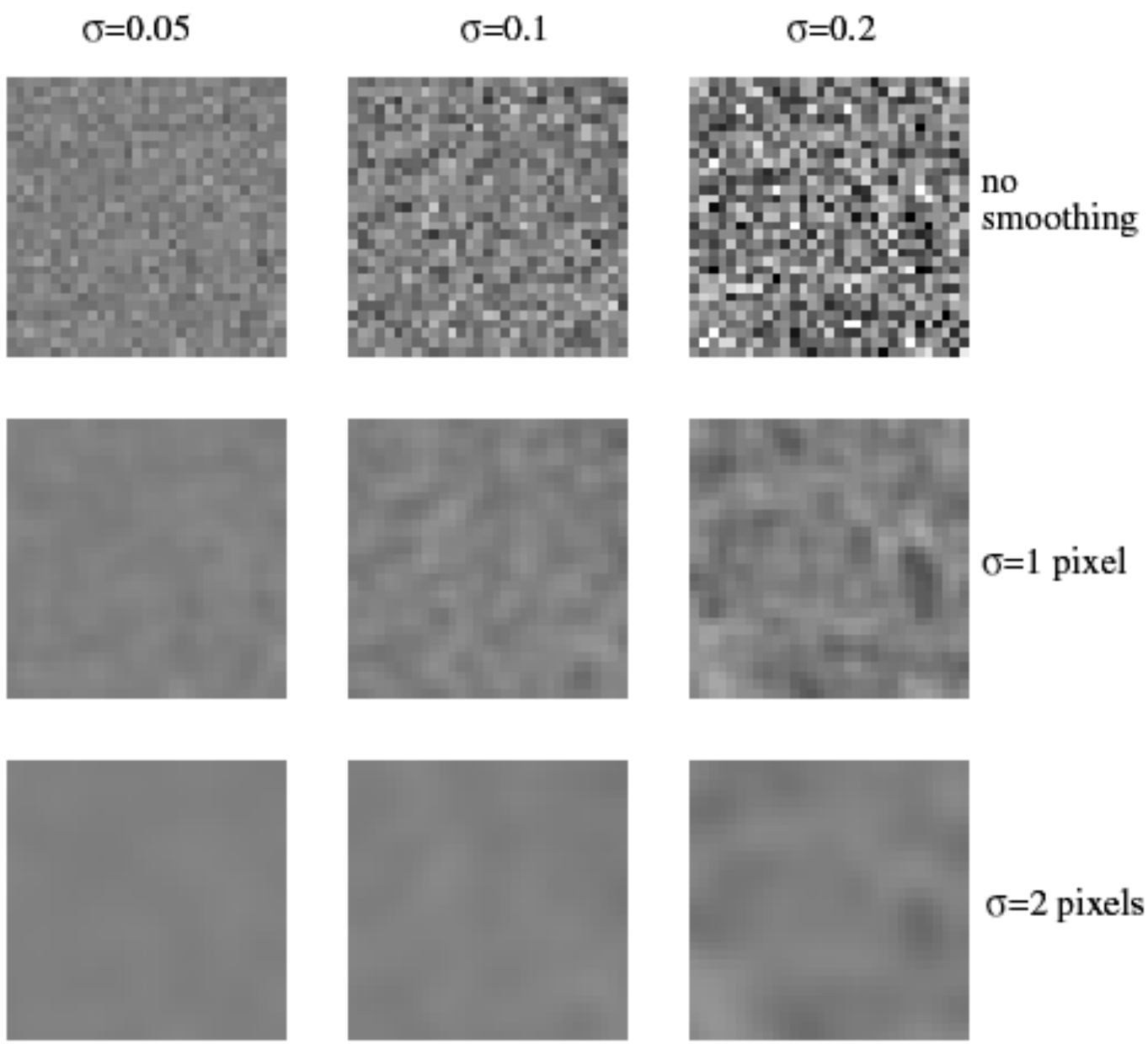


$\sigma = 2$
with 30×30 kernel



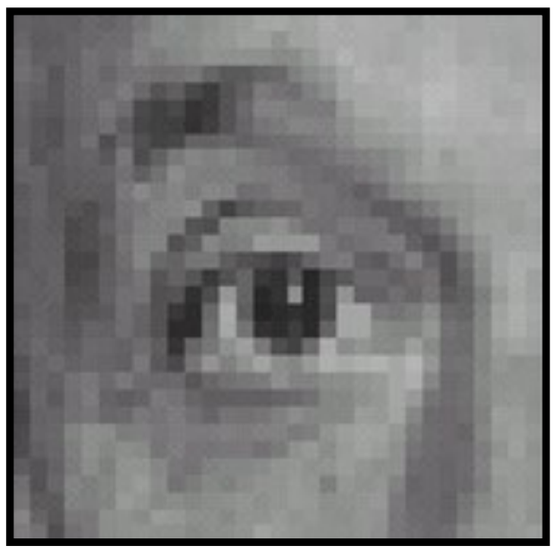
$\sigma = 5$
with 30×30 kernel

More noise →



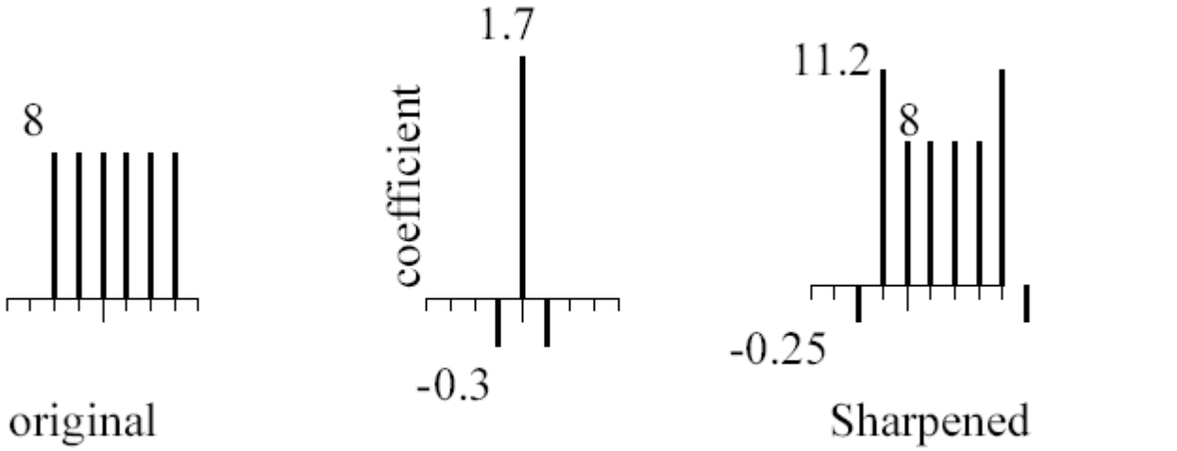
Wider smoothing kernel →

Linear Filters



Original

$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \frac{1}{9} \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

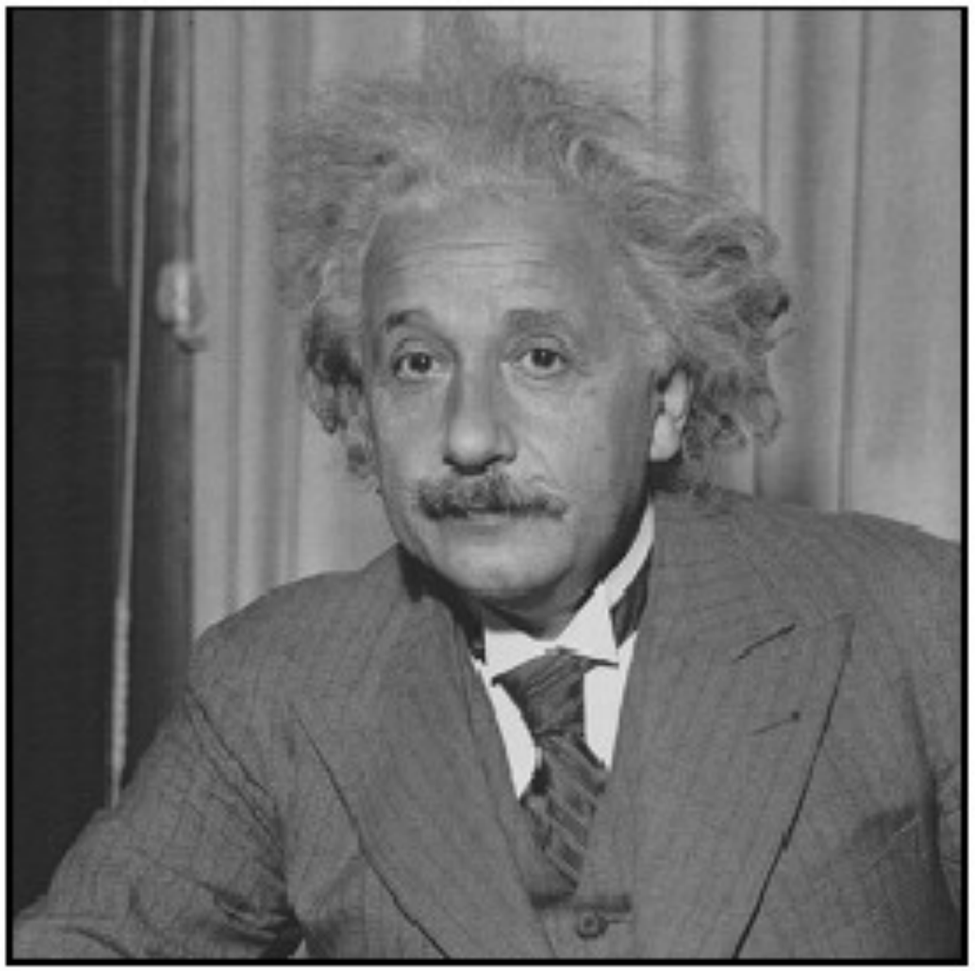


Sharpened (differences are accentuated; constant areas are left untouched).

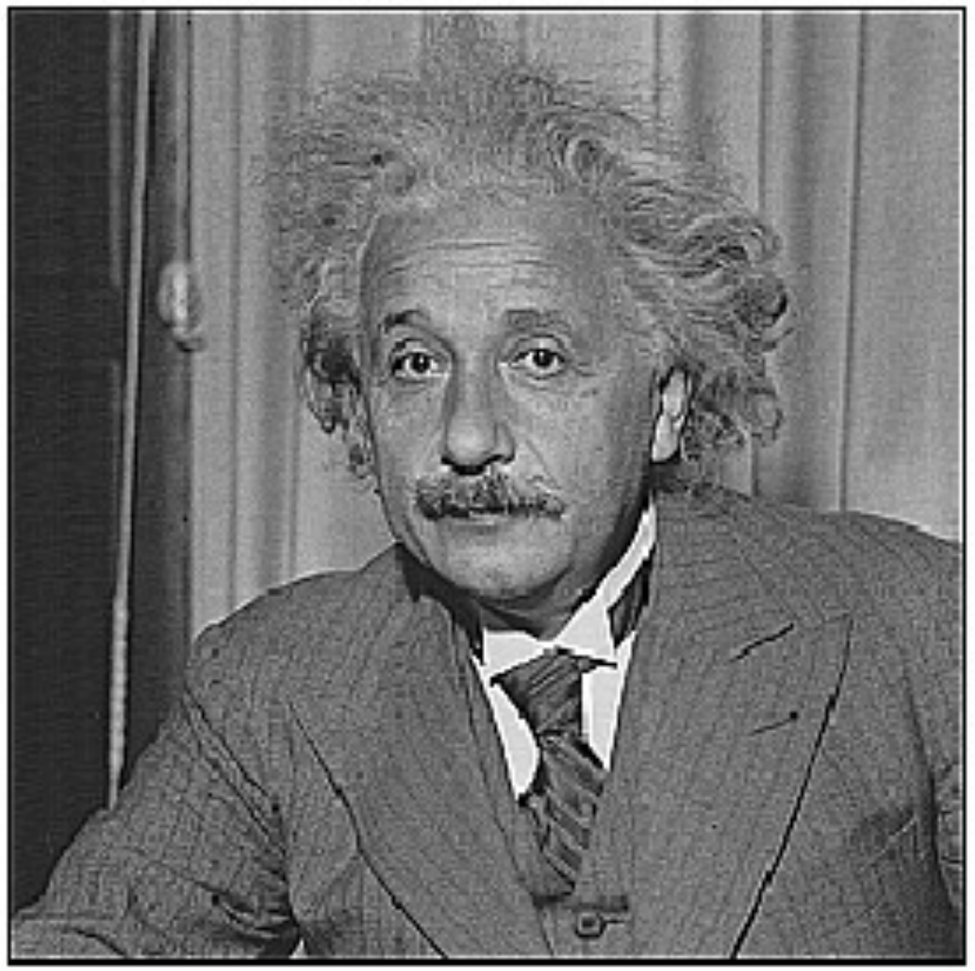
- Sharpening filter

Accentuates differences with local average

Filtering Examples: Sharpening



before



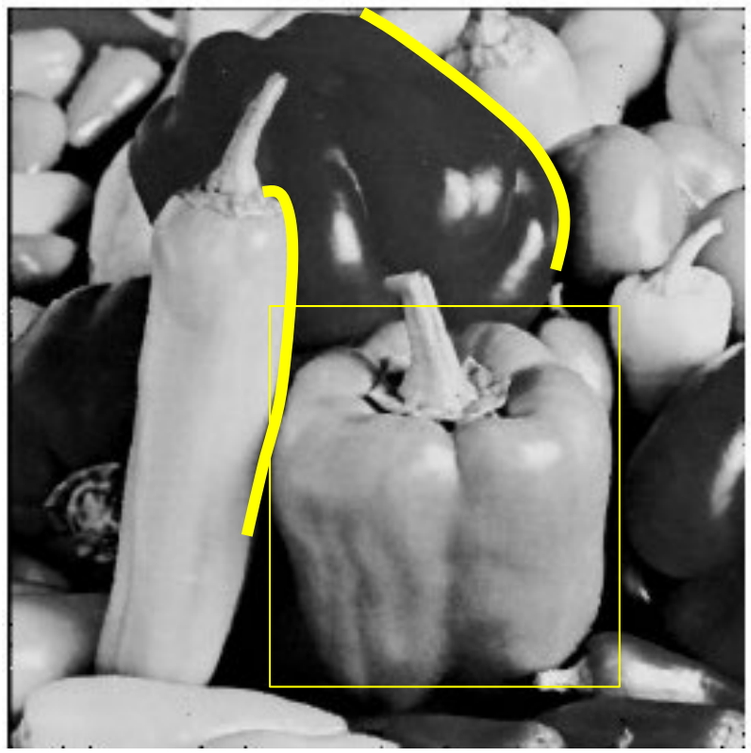
after

Filters for Features

- Previously, thinking of filtering as a way to remove or reduce noise

Filters for Features

- Previously, thinking of filtering as a way to remove or reduce noise
- Now, consider how filters will allow us to **abstract** higher-level "**features**".
- Map **raw pixels** to an **intermediate representation** that will be used for subsequent processing



Filters for Features

- Previously, thinking of filtering as a way to remove or reduce noise
- Now, consider how filters will allow us to **abstract** higher-level "**features**".
- Map **raw pixels** to an **intermediate representation** that will be used for subsequent processing
- **Goal: reduce amount of data, discard redundancy, preserve what's useful**