

Lecture 2:

# Image Formation, Basic Image Processing

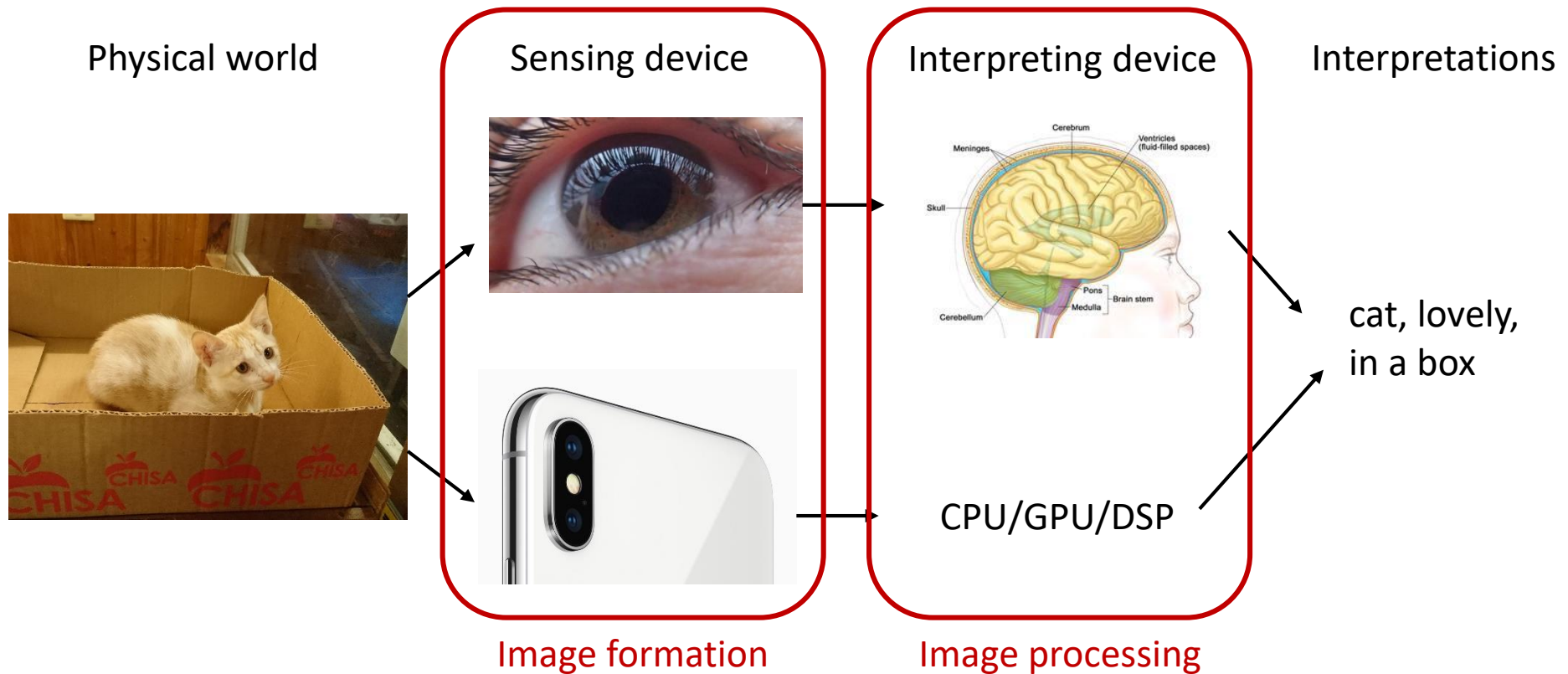
Wei-Chih Tu (塗偉志)

National Taiwan University

Fall 2018

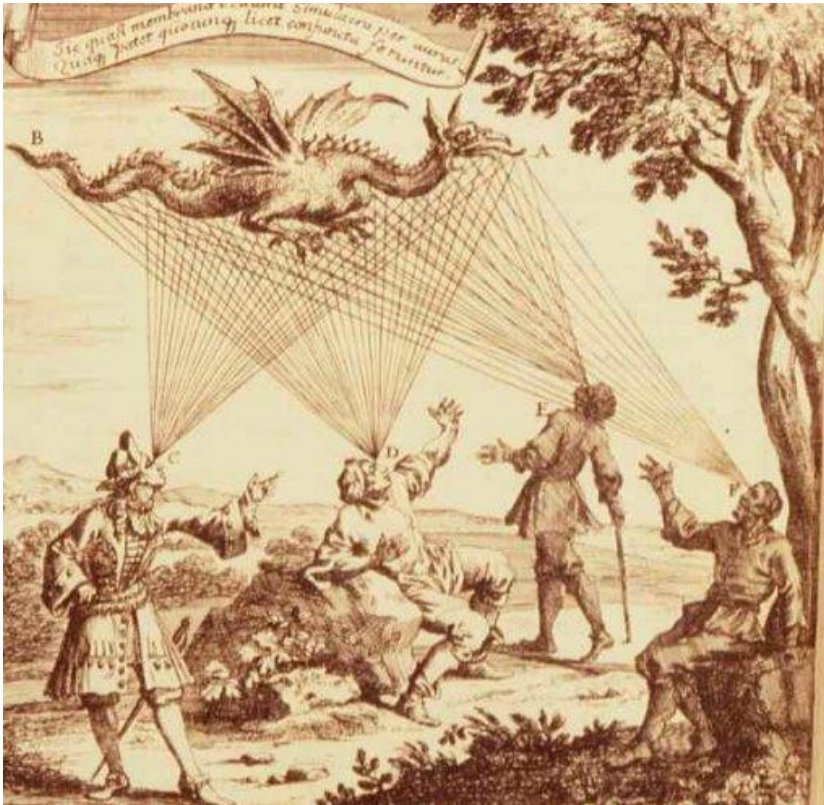
# Vision

- How vision is formed



# Image Formation

- Emission theory of vision



Eyes send out “feeling rays” into the world

Supported by:

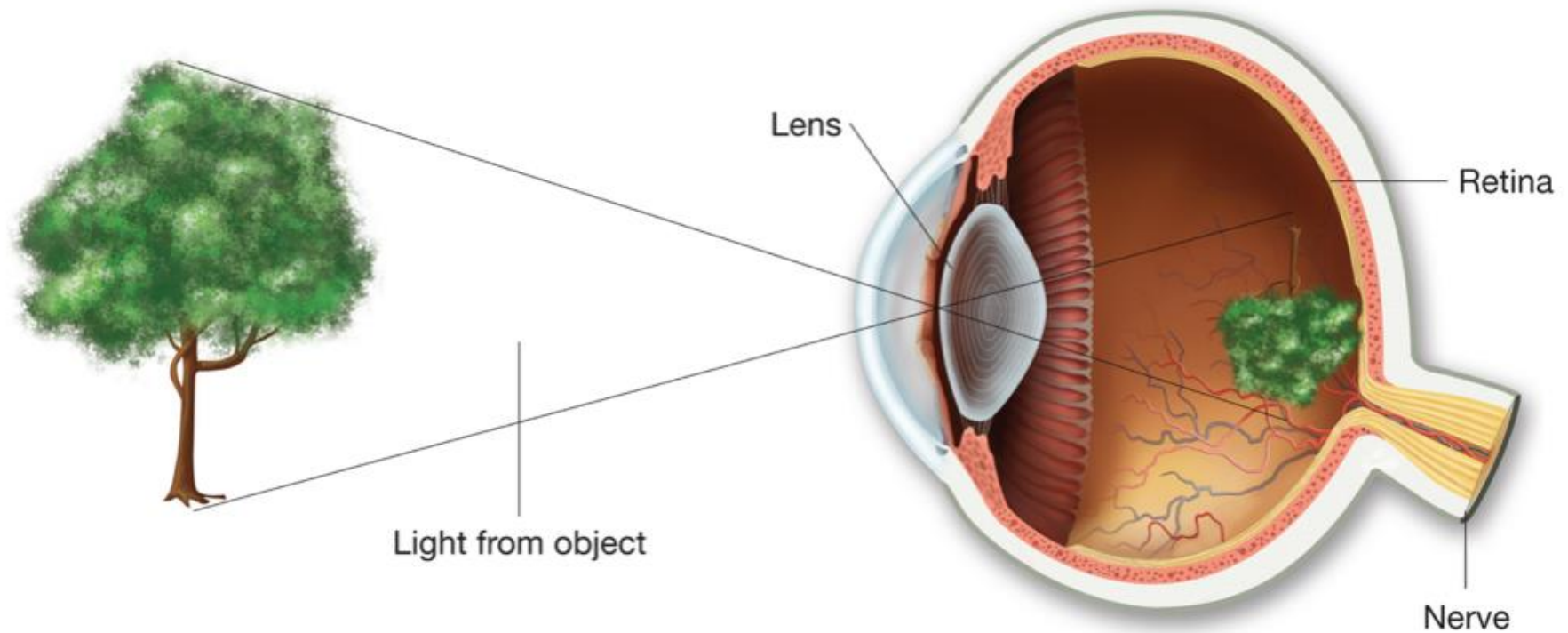
- Empedocles
- Plato
- Euclid (kinda)
- Ptolemy
- ...
- 50% of US college students\*

[\\*http://www.ncbi.nlm.nih.gov/pubmed/12094435?dopt=Abstract](http://www.ncbi.nlm.nih.gov/pubmed/12094435?dopt=Abstract)

“For every complex problem there is an answer that is clear, simple, and wrong.”  
-- H. L. Mencken

# Image Formation

- The human eye is a camera
  - The image is inverted, but the spatial relationships are preserved



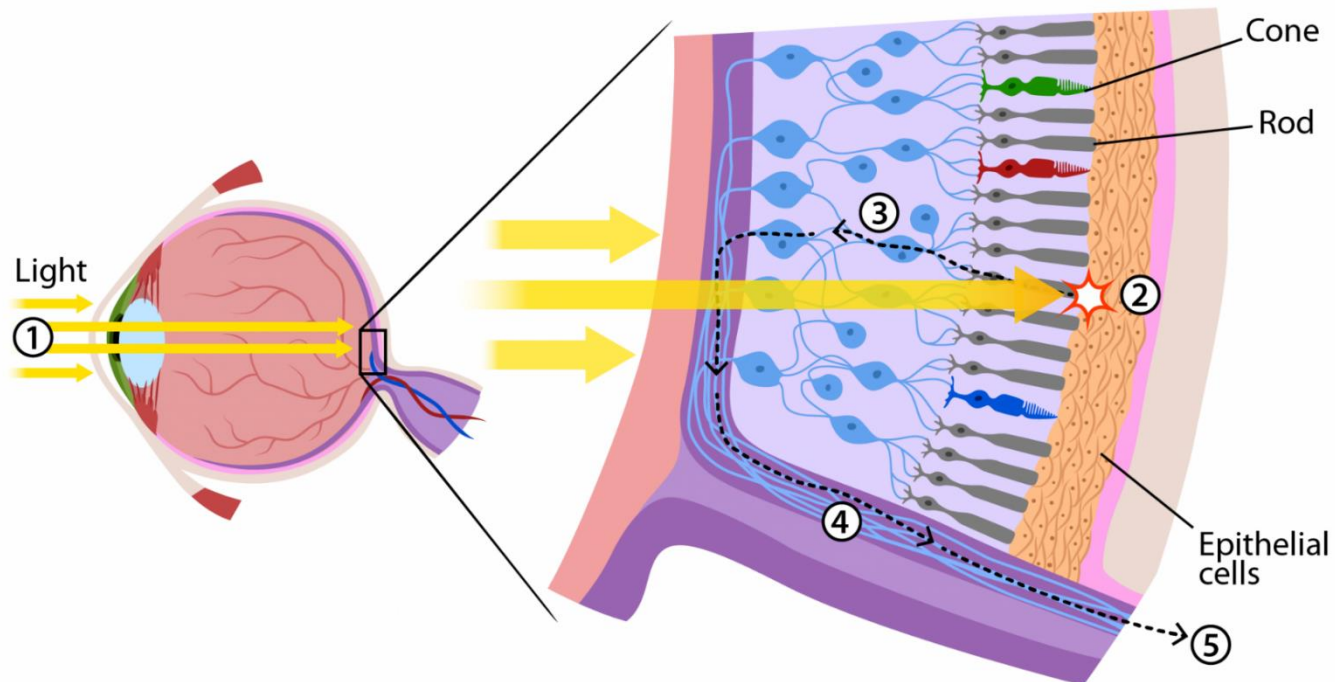
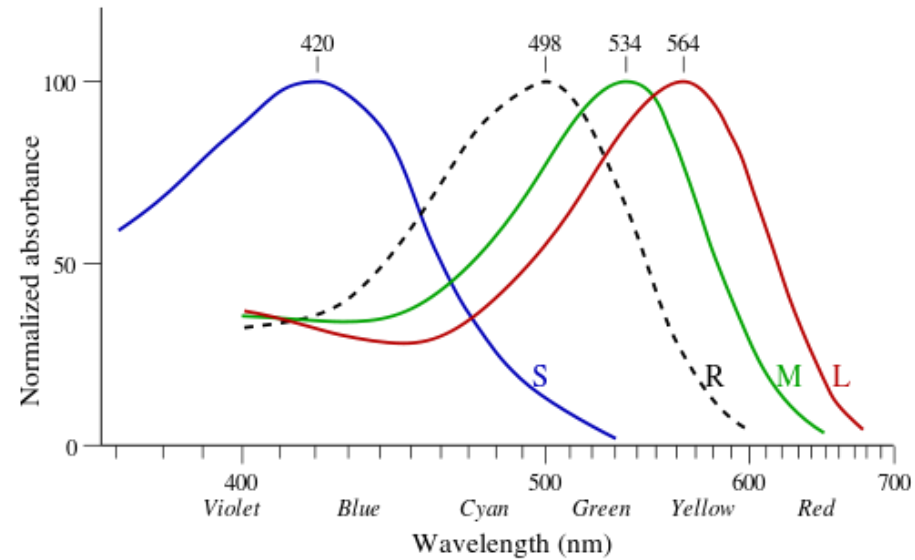
# Mini-tutorial on Color

- What is color?
  - **Def** *The property possessed by an object of producing different **sensations** on the eye as a result of the way it reflects or emits light.*  
-- Oxford Dictionary
- Color is perceptual!
  - Which one is “true blue”?



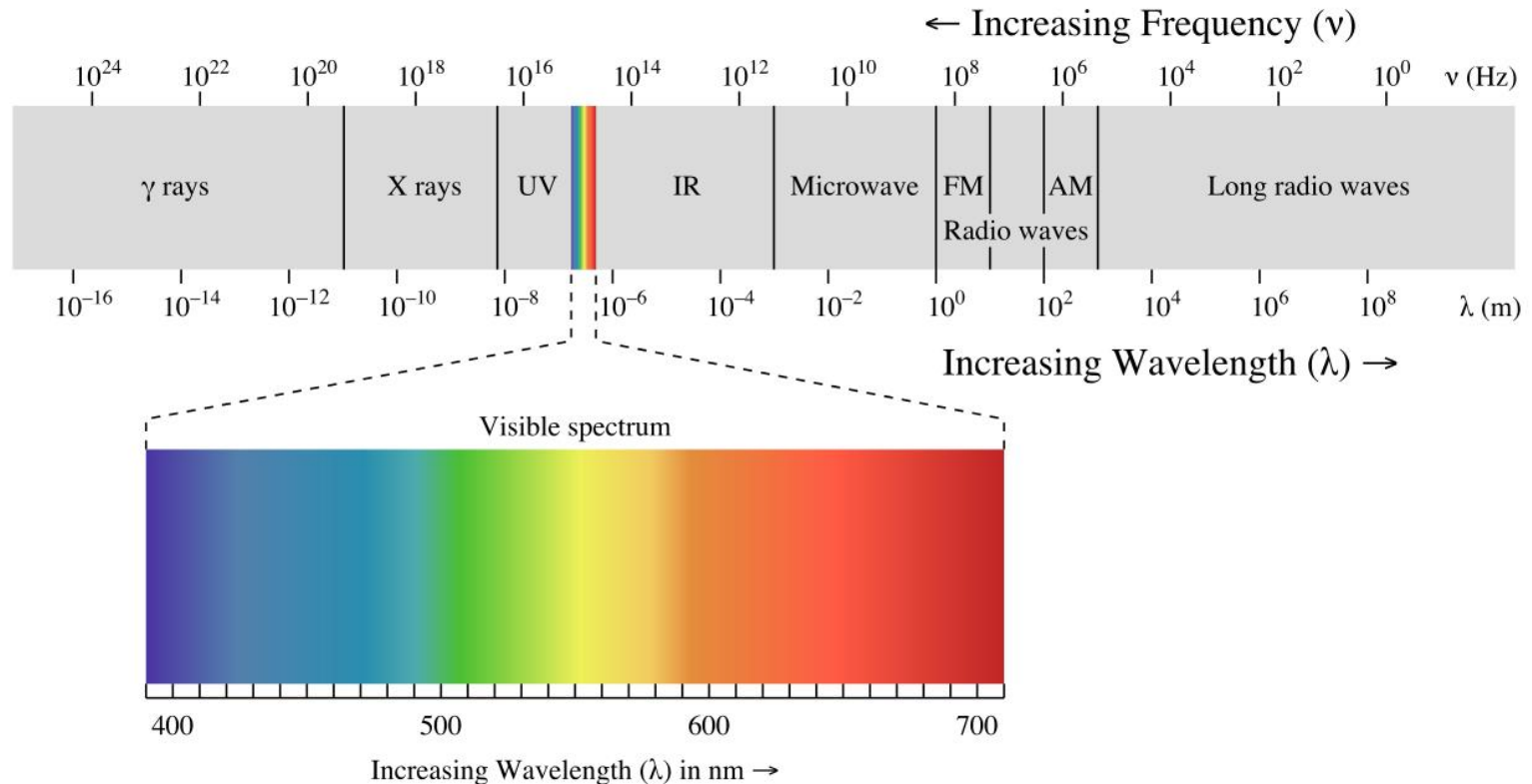
# Color

- How do we see light?
  - Cones (short, medium, long)
  - Rods



# Color

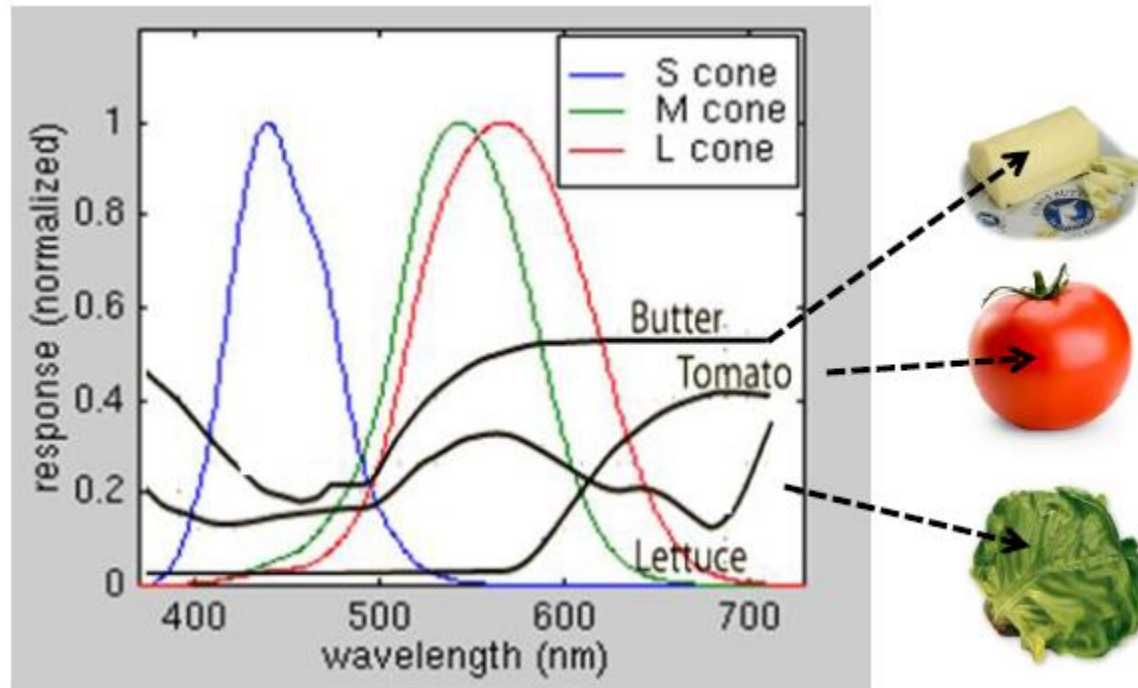
- We only see very small range of EM spectrum





# Color

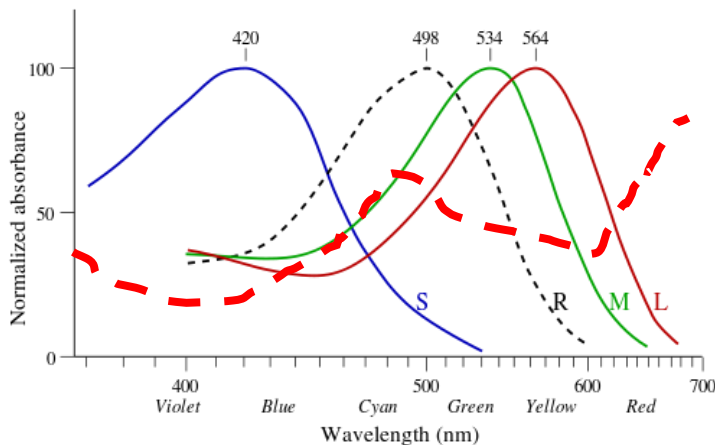
- Spectral power distribution (SPD)





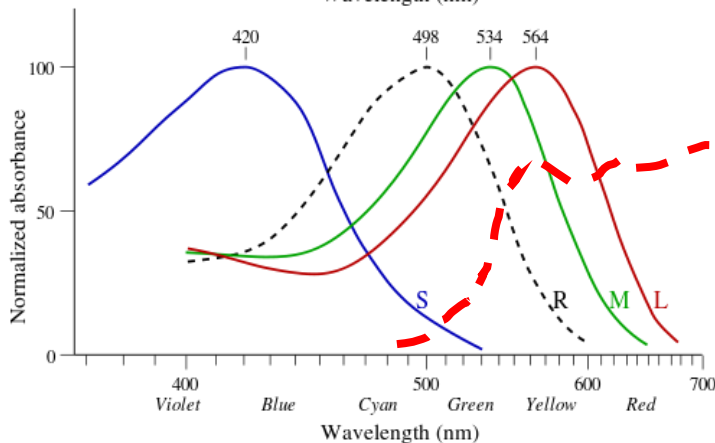
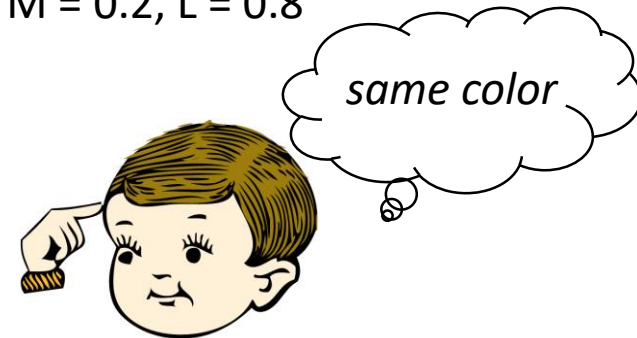
# Color

- Human vision sensation is a “dimension reduction”!
  - Many-to-one mapping



SPD of a real apple

→ stimulating  $S = 0.2$ ,  $M = 0.2$ ,  $L = 0.8$



SPD of red ink

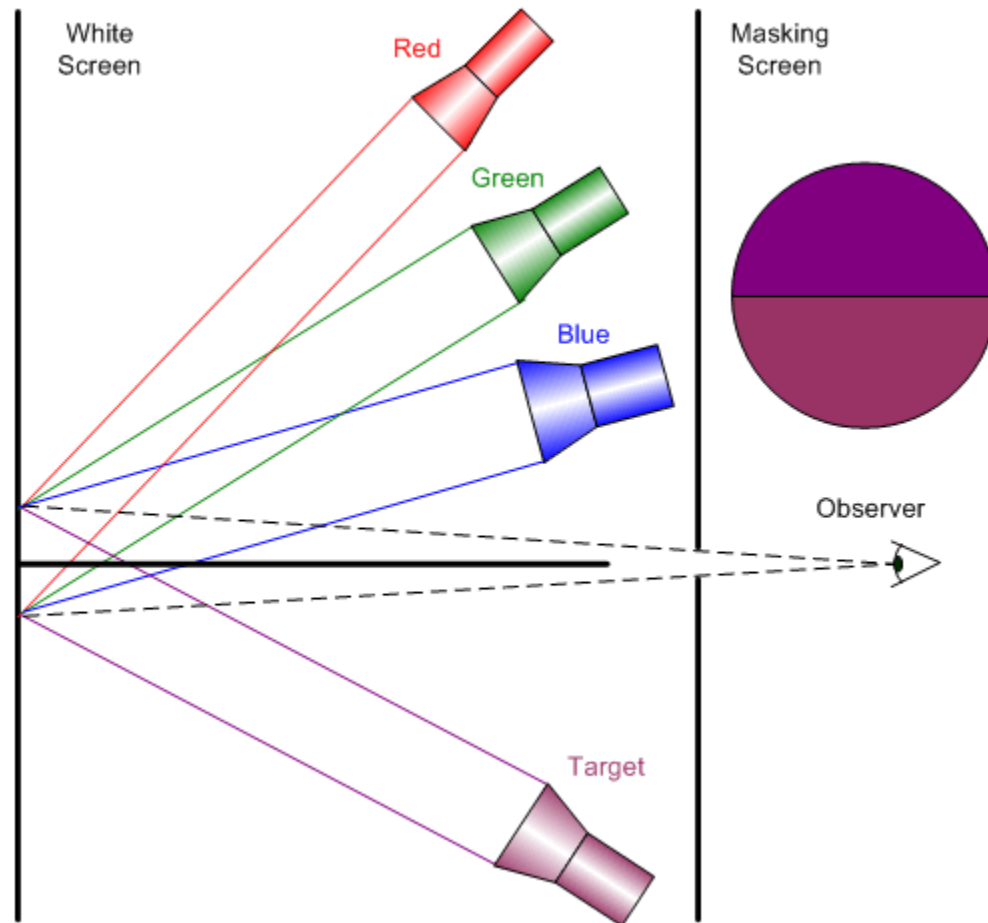
→ stimulating  $S = 0.2$ ,  $M = 0.2$ ,  $L = 0.8$

# Color

- Tristimulus color theory (1850~)
- Grassman's Law
  - A source color can be matched by a linear combination of three independent primaries.
- Quantifying color
  - SPDs go through a “black box” (human visual system) and are perceived as color
  - Scientists performed human study...

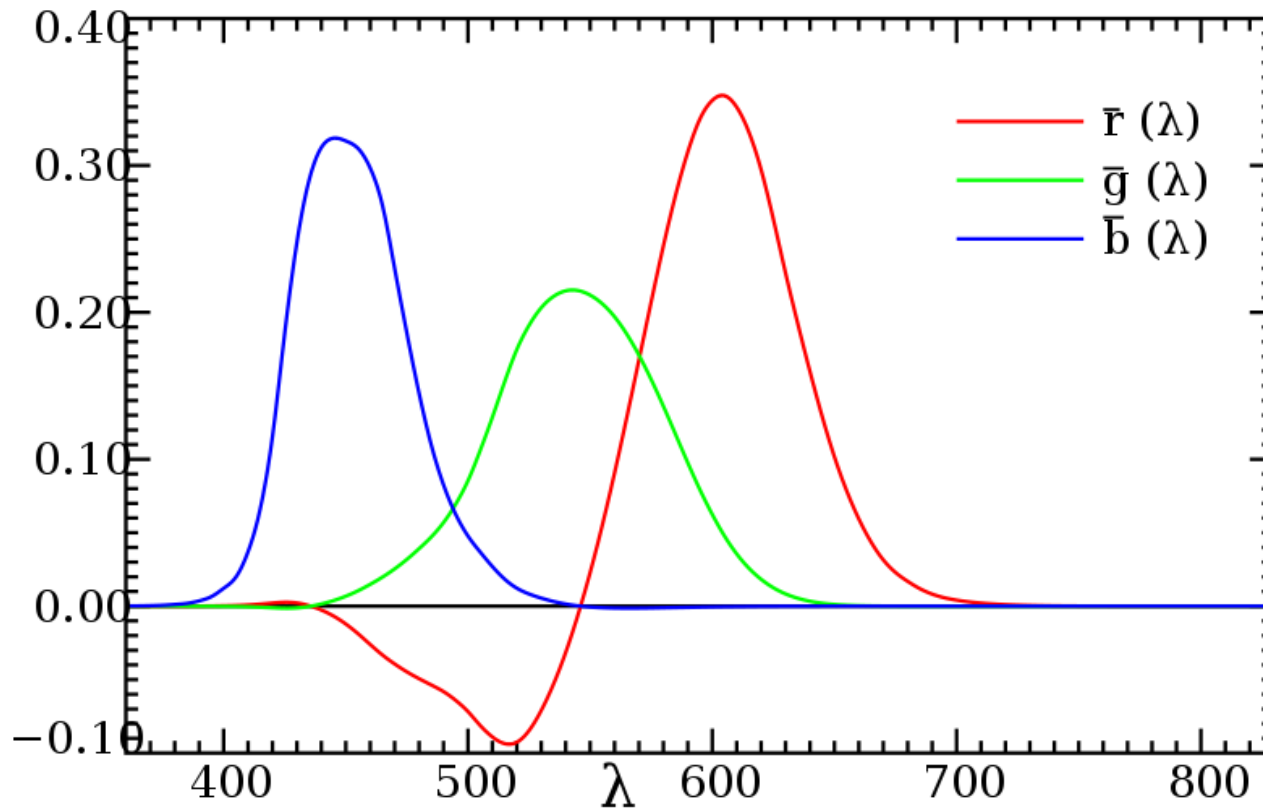
# Color

- CIE RGB color matching



# Color

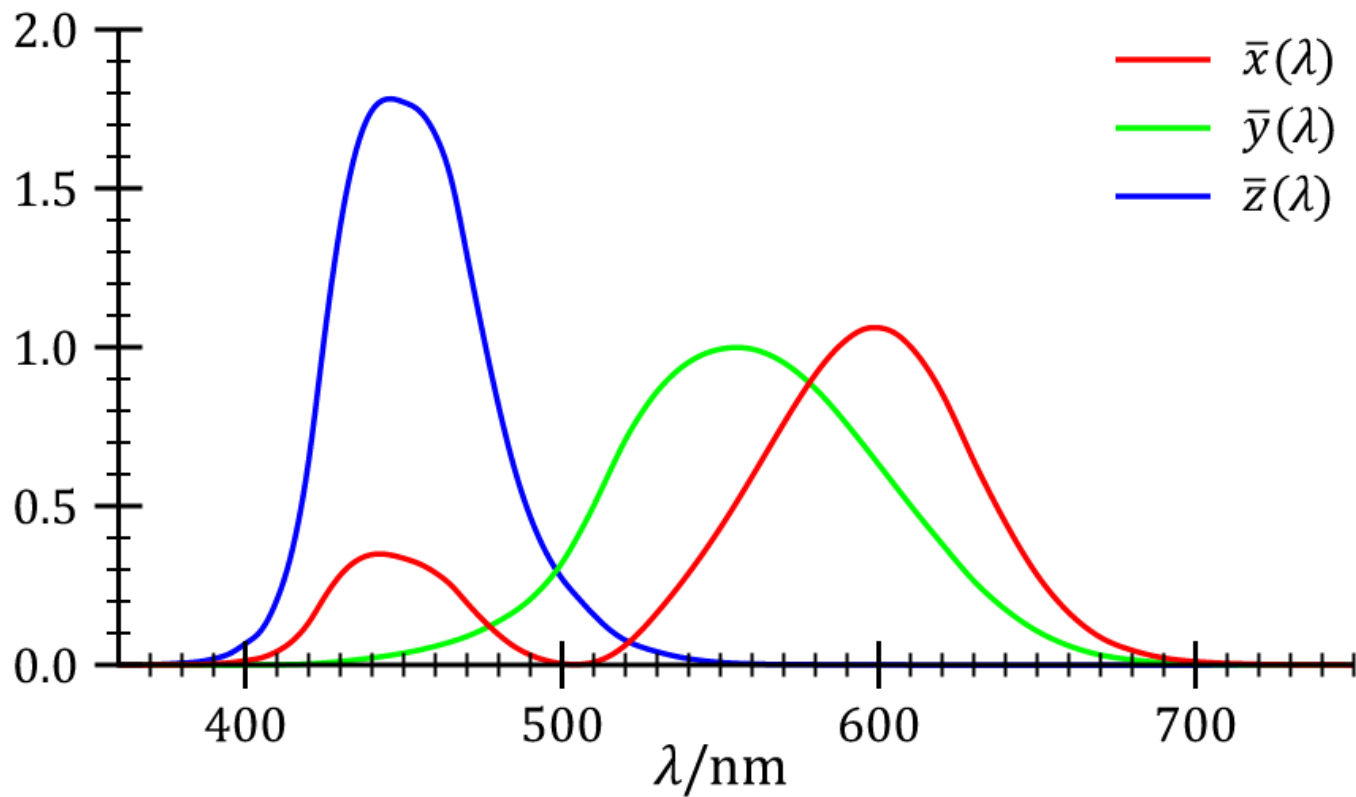
- CIE RGB results



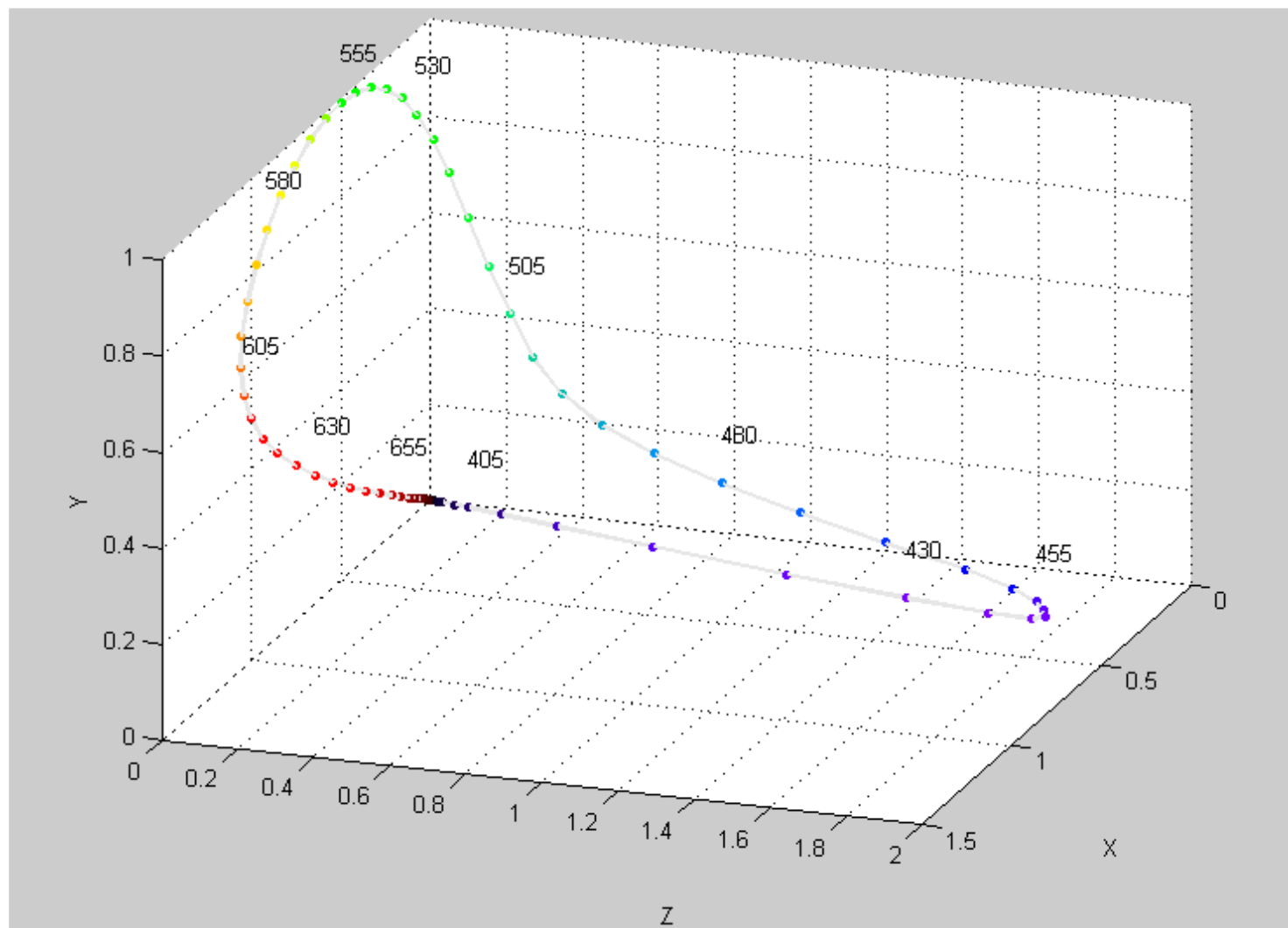
# Color

- CIE 1931

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \frac{1}{b_{21}} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \frac{1}{0.176,97} \begin{bmatrix} 0.490,00 & 0.310,00 & 0.200,00 \\ 0.176,97 & 0.812,40 & 0.010,630 \\ 0.000,0 & 0.010,000 & 0.990,00 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$



# CIE XYZ 3D Plot



# Color

- What does it mean?
  - Given a SPD,  $I(\lambda)$ , we can find its mapping into the CIE XYZ space

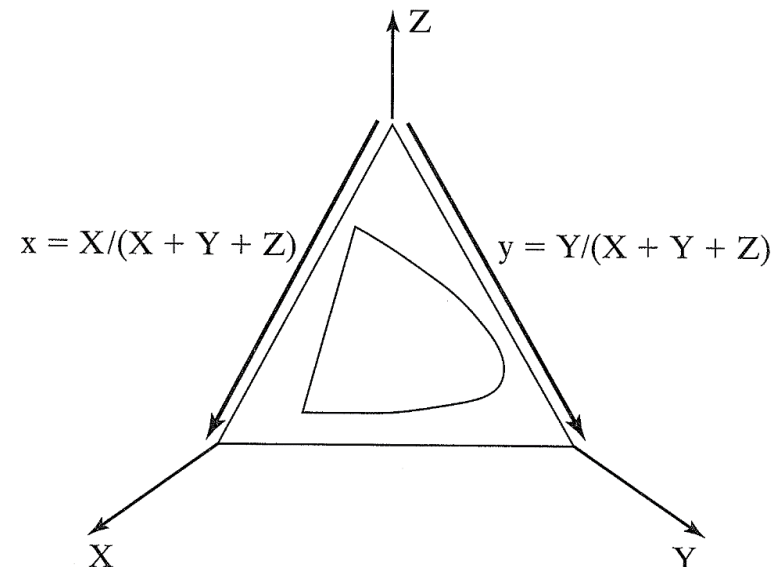
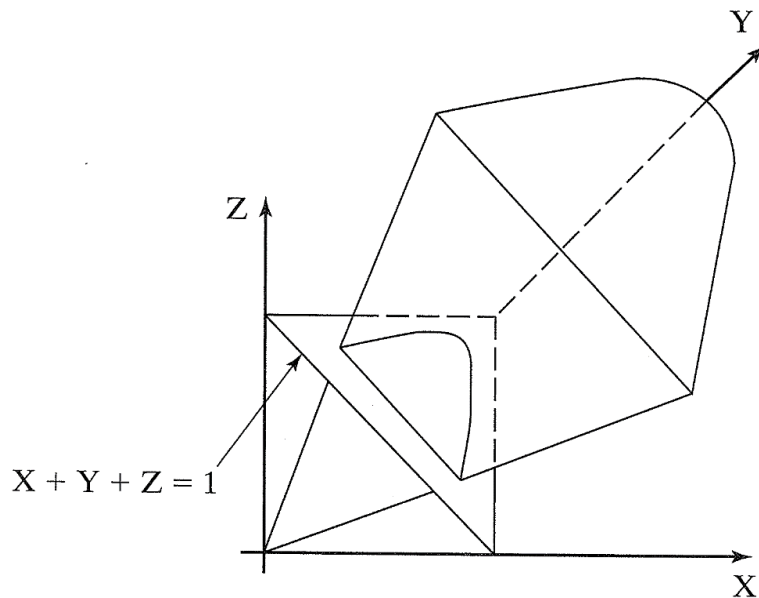
$$X = \int_{380}^{780} I(\lambda) \bar{x}(\lambda) d\lambda \quad Y = \int_{380}^{780} I(\lambda) \bar{y}(\lambda) d\lambda \quad Z = \int_{380}^{780} I(\lambda) \bar{z}(\lambda) d\lambda$$

- Given two SPDs, if their CIE XYZ values are equal, they are considered the same perceived color



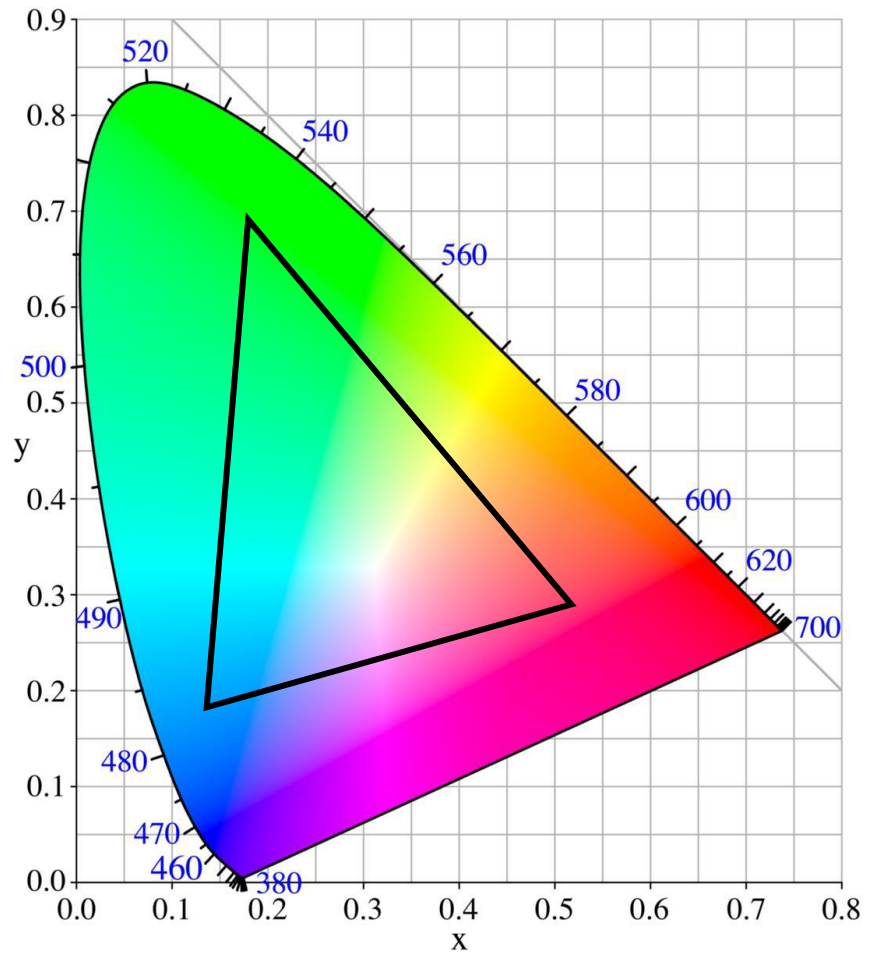
# Color

- Sometimes it is useful to discuss color in terms of luminance and chromaticity
- CIE xy chromaticity is designed for this purpose
  - Project the CIE XYZ values onto the  $X + Y + Z = 1$  plane



# Color

- CIE xy chromaticity diagram
  - Gamut

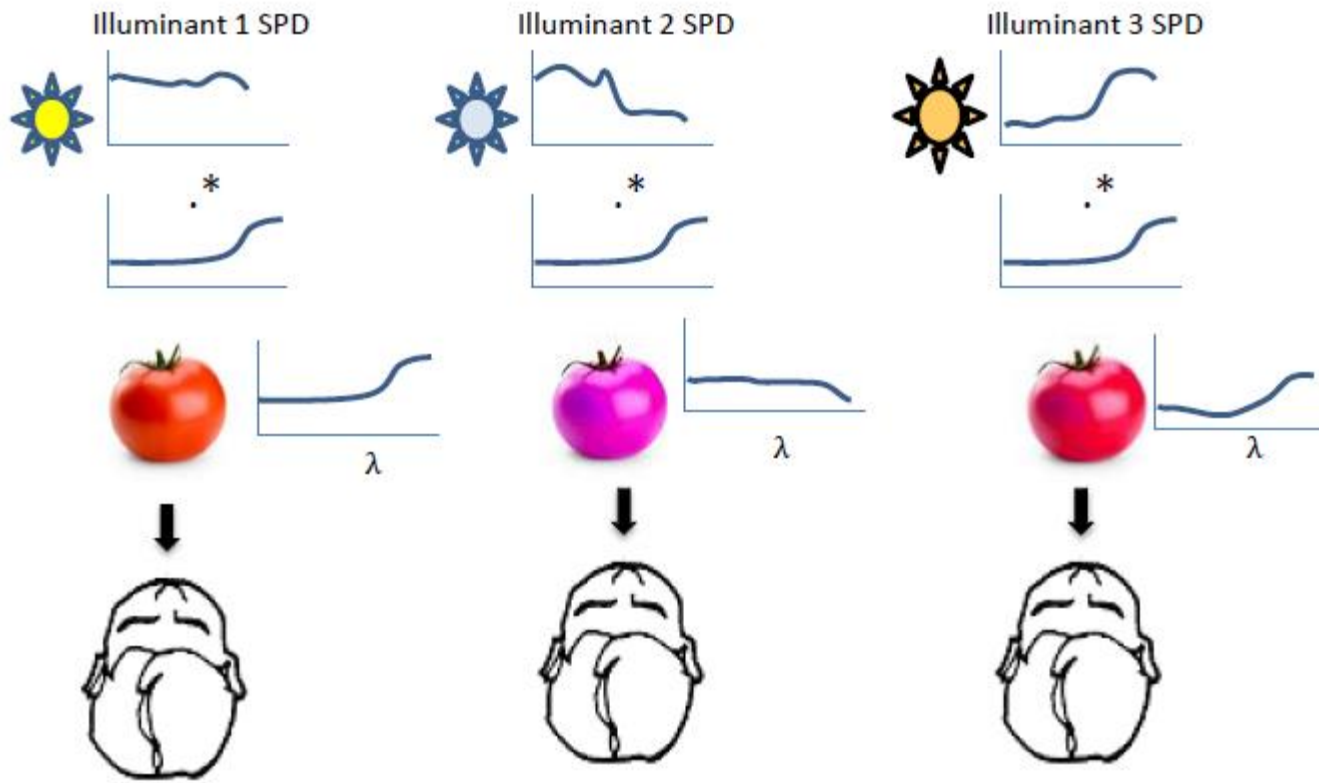


# Color

- The XYZ values are **not** specific to any device
- What does RGB = (33, 55, 128) mean?
  - For example: iPhone Xs vs. HTC U12+
- Devices (camera, scanners, displays, ...) can find mappings of their specific values to corresponding CIE XYZ values
  - Standard RGB (sRGB)
  - This provides a canonical space to match between devices!

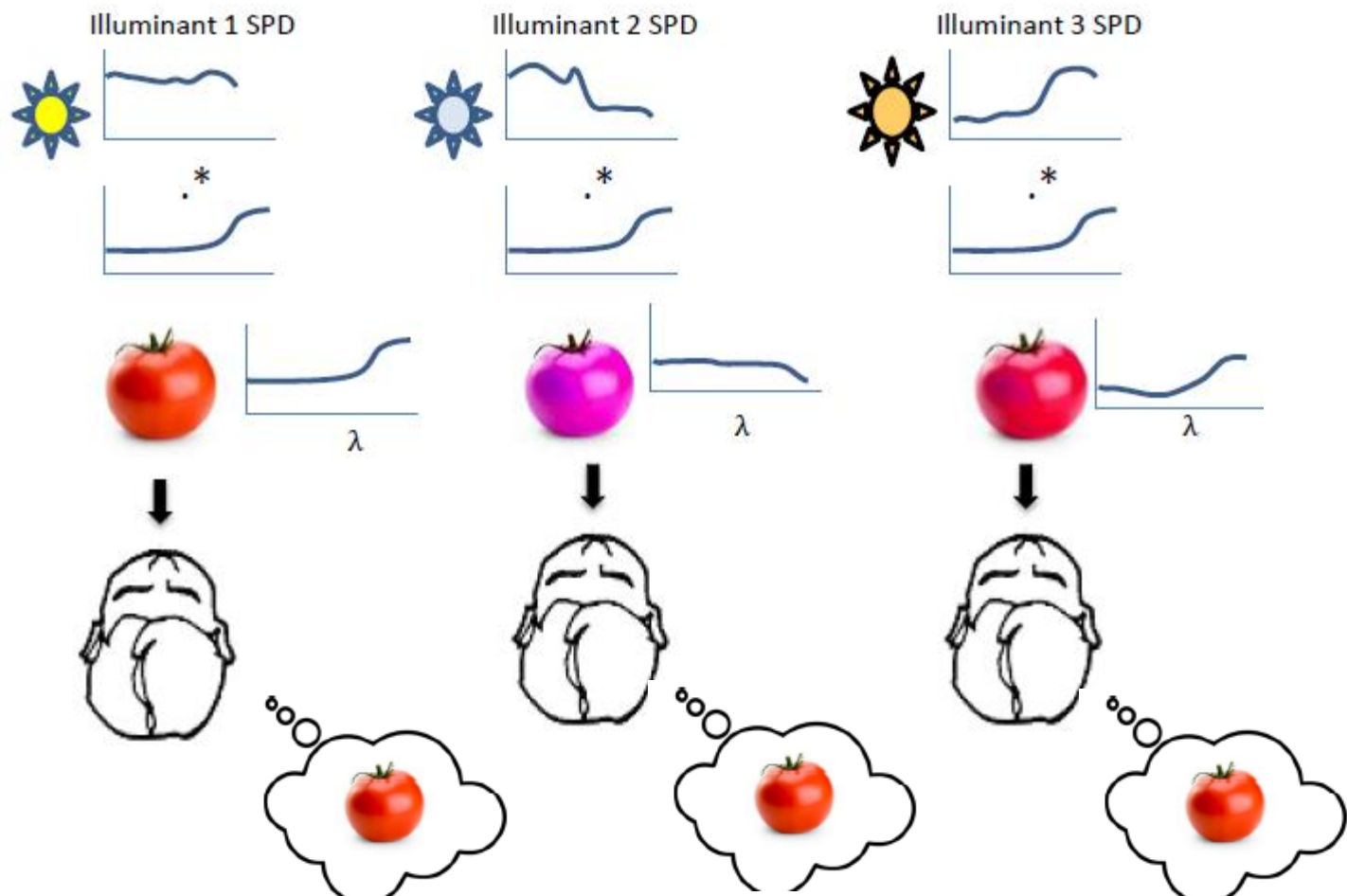
# Color

- And the light source matters



# Color

- Color constancy: They “perceive” the same!



# Color

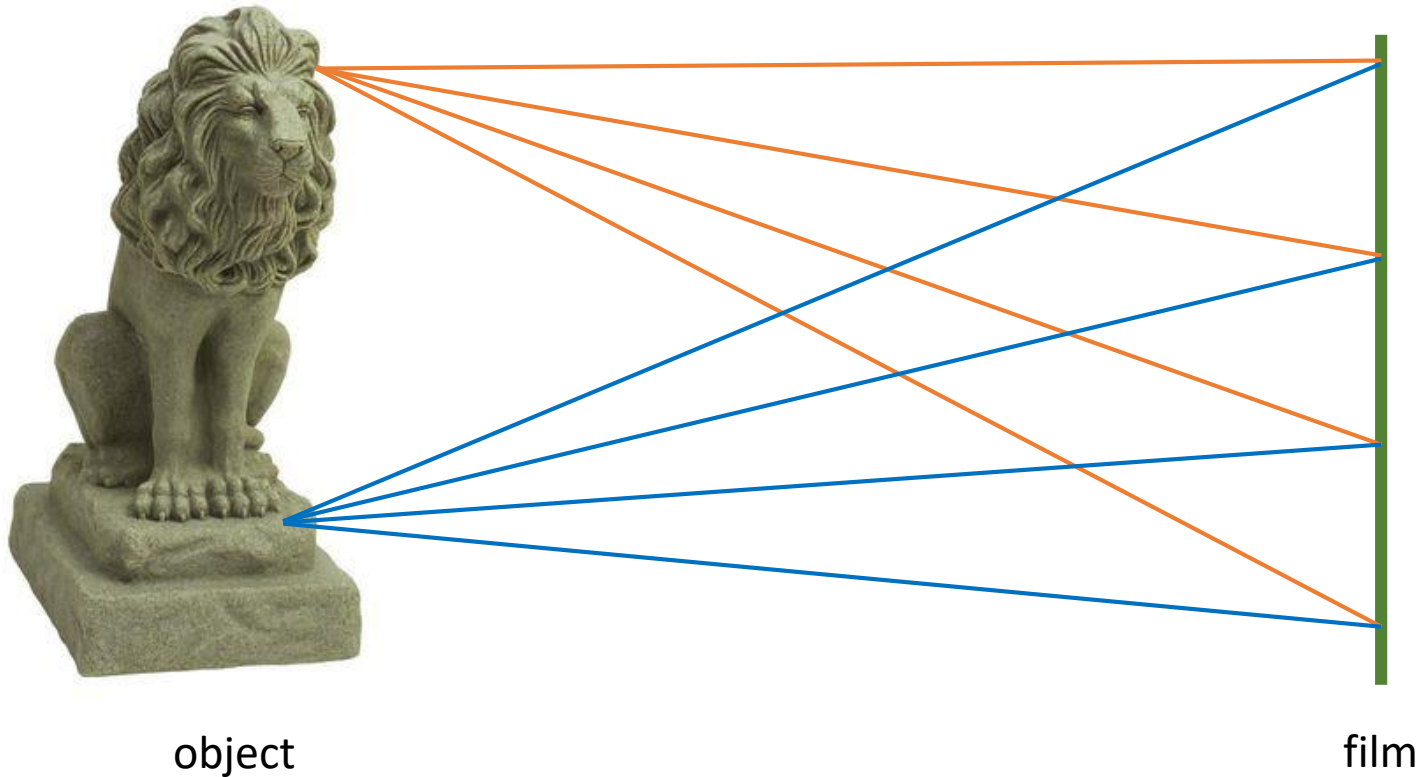
- More color constancy



[https://motherboard.vice.com/en\\_us/article/this-picture-has-no-red-pixelsso-why-do-the-strawberries-still-look-red](https://motherboard.vice.com/en_us/article/this-picture-has-no-red-pixelsso-why-do-the-strawberries-still-look-red)

# Image Formation

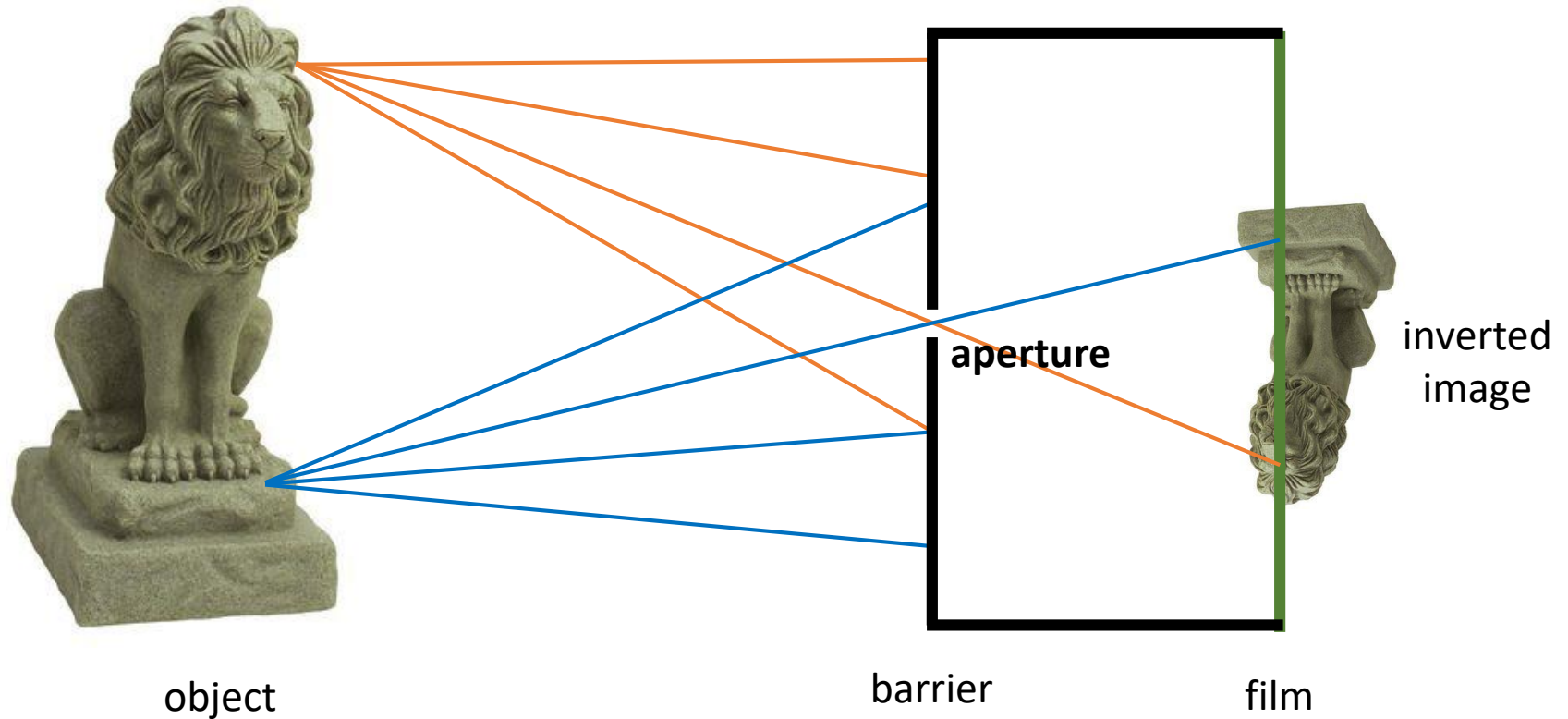
- Building a camera
  - Put a piece of film in front of an object





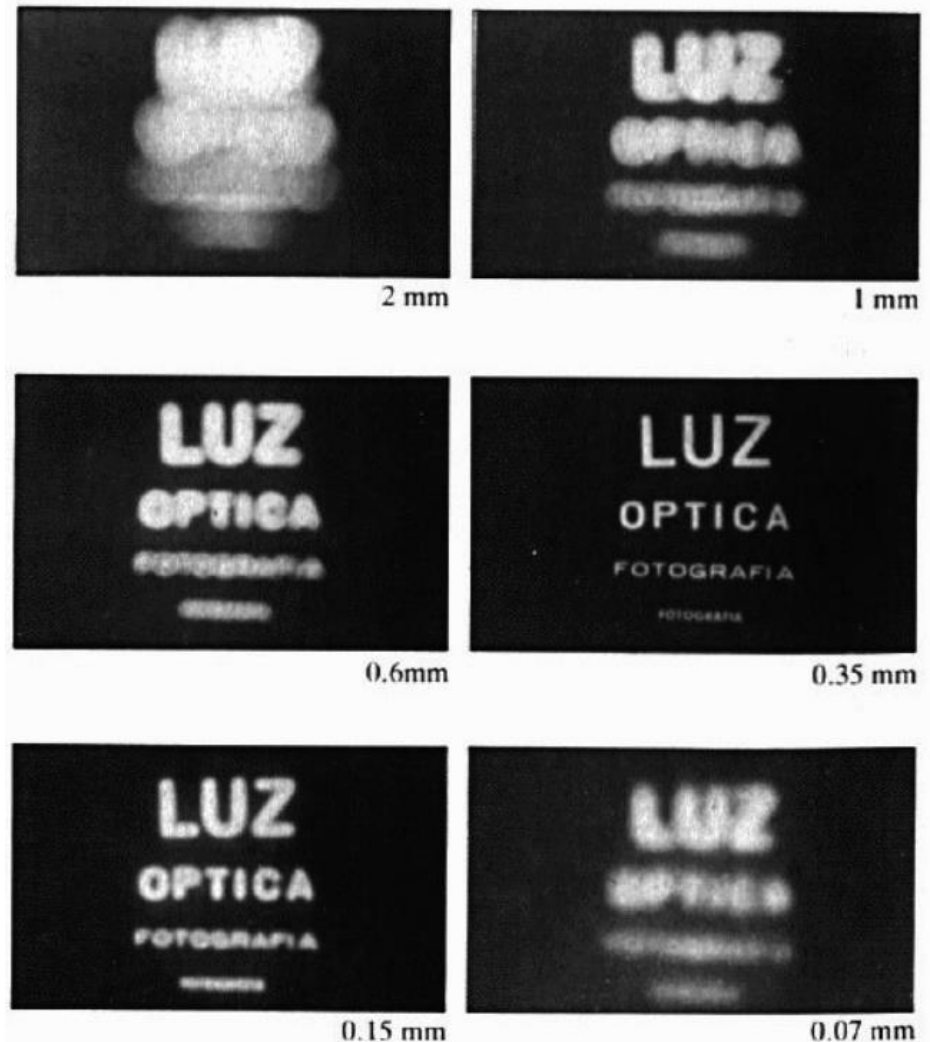
# Image Formation

- Add a barrier to block off most of the rays
  - This reduces blurring



# Aperture Size Matters

- Why not making the aperture as small as possible?
  - Less light get through
  - Diffraction effect



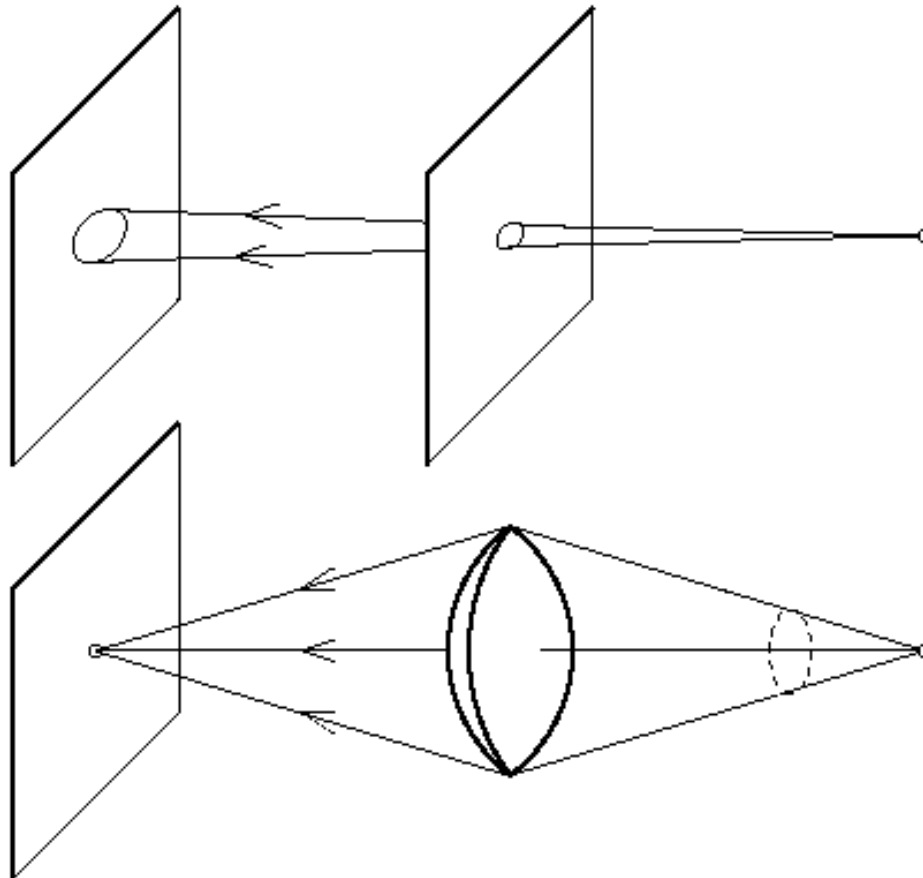
# The “Trashcam” Project



<https://petapixel.com/2012/04/18/german-garbage-men-turn-dumpsters-into-giant-pinhole-cameras/>

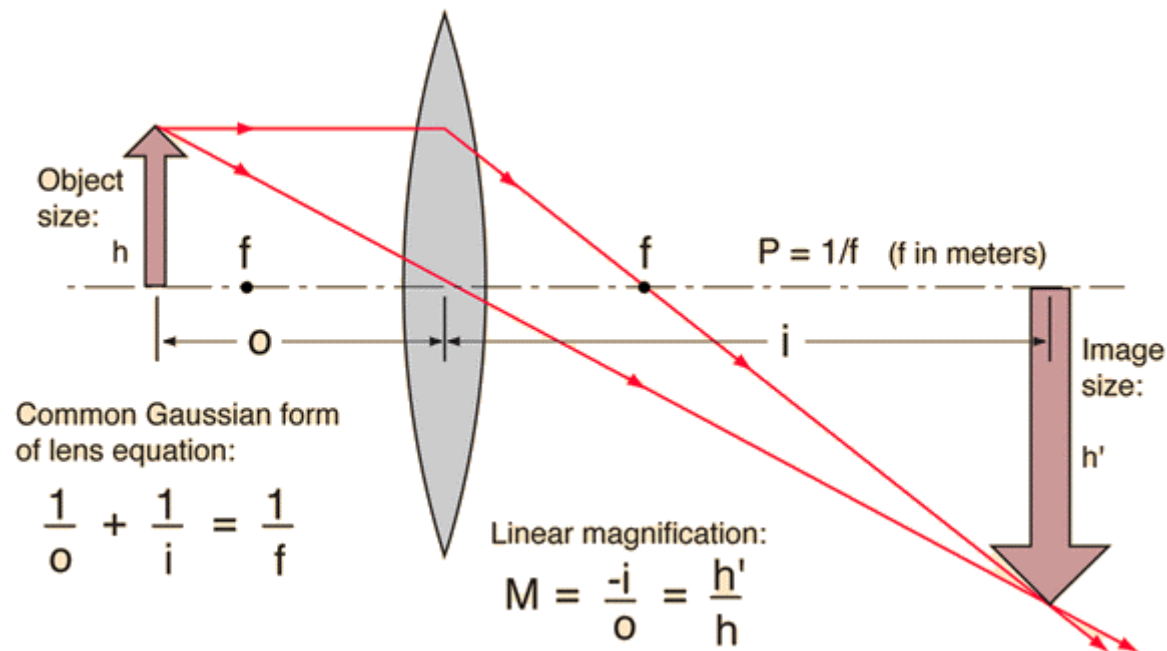
# Image Formation

- Adding a lens



# Image Formation

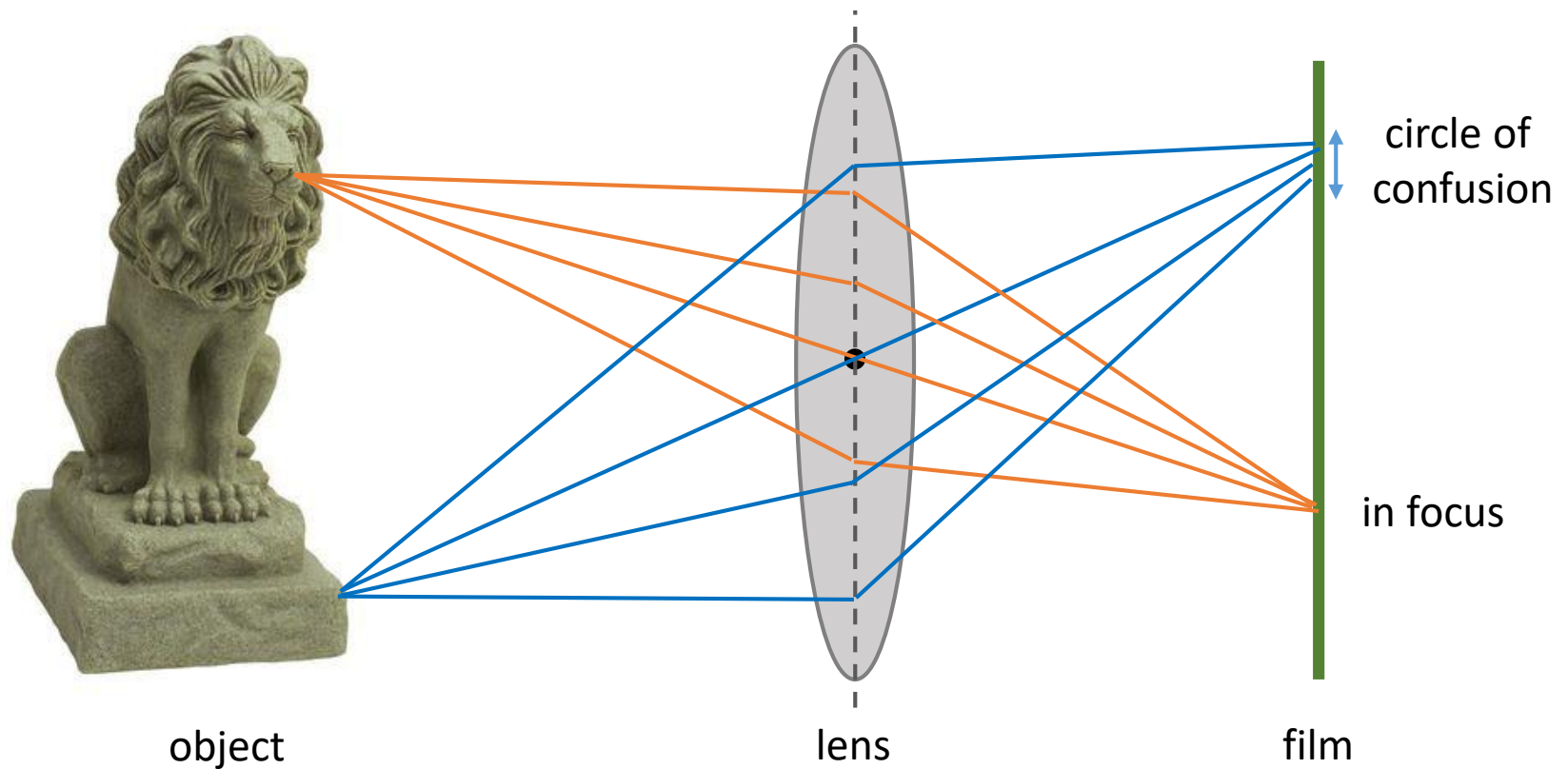
- Thin lens equation



Source: <https://www.chegg.com/homework-help/questions-and-answers/theory-thin-lens-equation-written-1-f-1-o-1-f-focal-length-o-object-distance-image-distanc-q13090621>

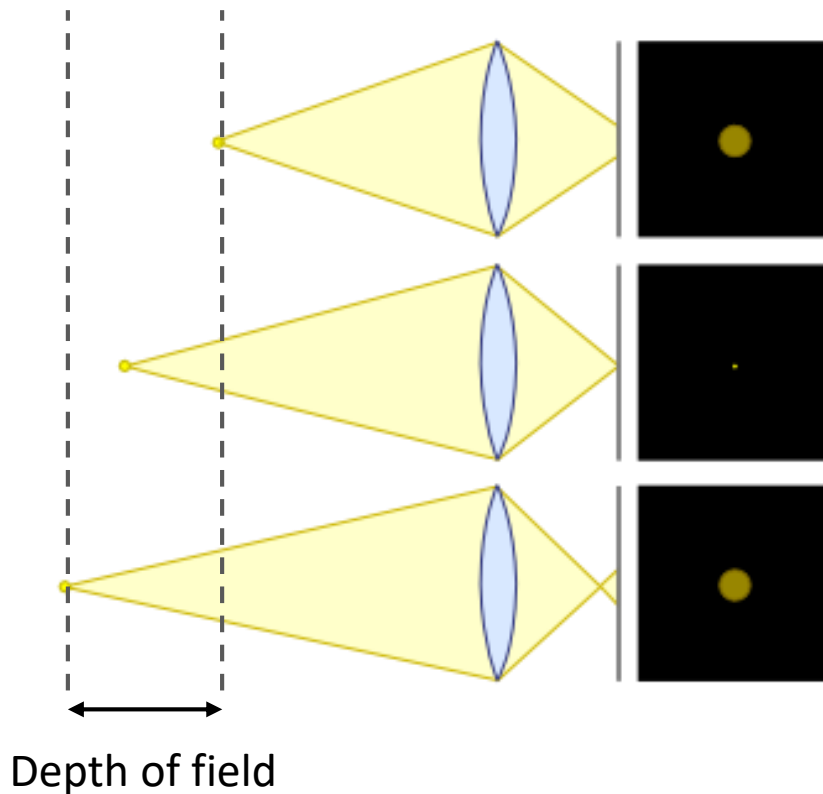
# Image Formation

- The lens focuses light onto the film



# Image Formation

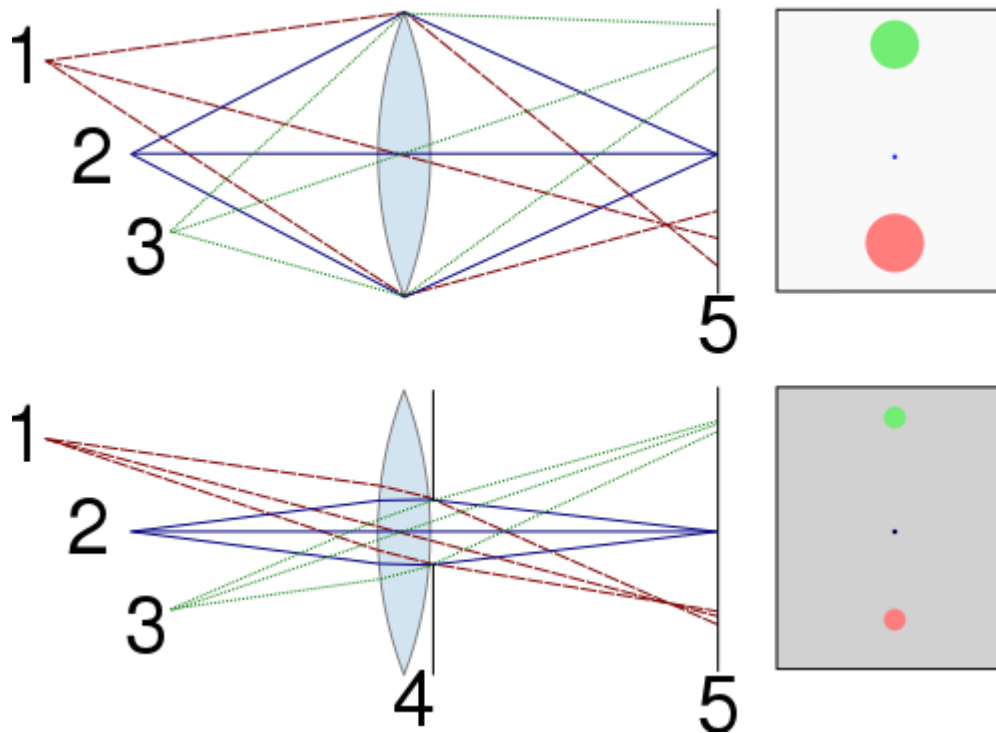
- Circle of confusion controls depth of field





# Image Formation

- Aperture also controls depth of field



# Image Formation

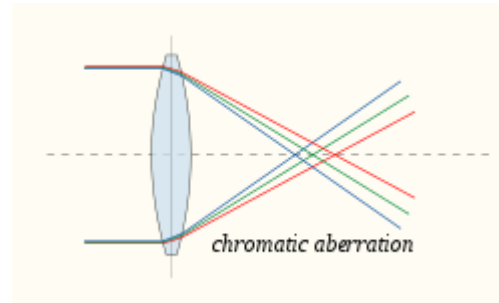
- Defocus



Source: AMC

# Image Formation

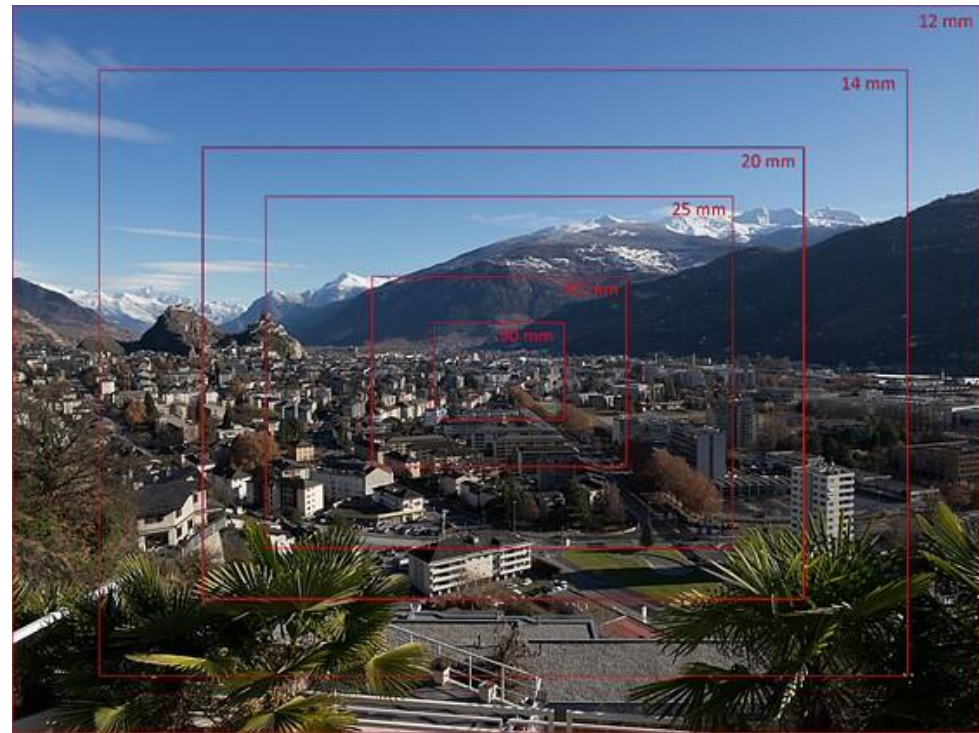
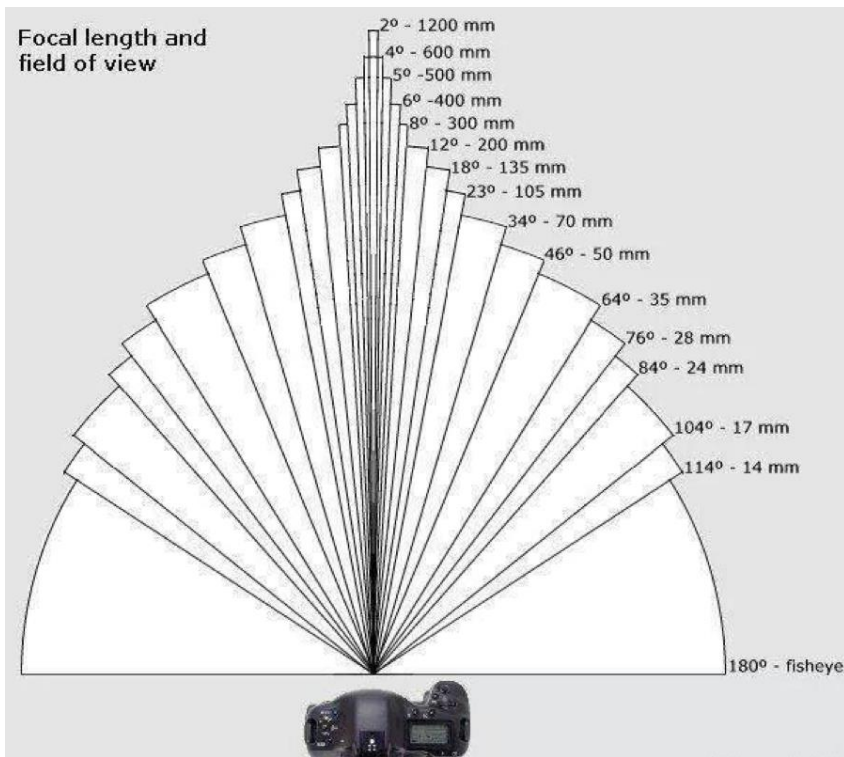
- Real lens consists of two or more pieces of glass
  - To alleviate chromatic aberration and vignetting



Vignetting

# Image Formation

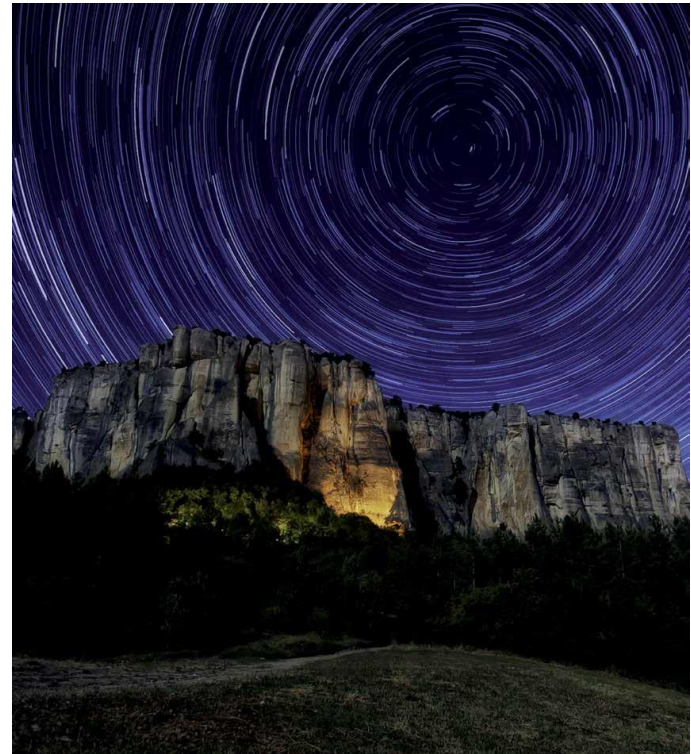
- Focal length controls field of view



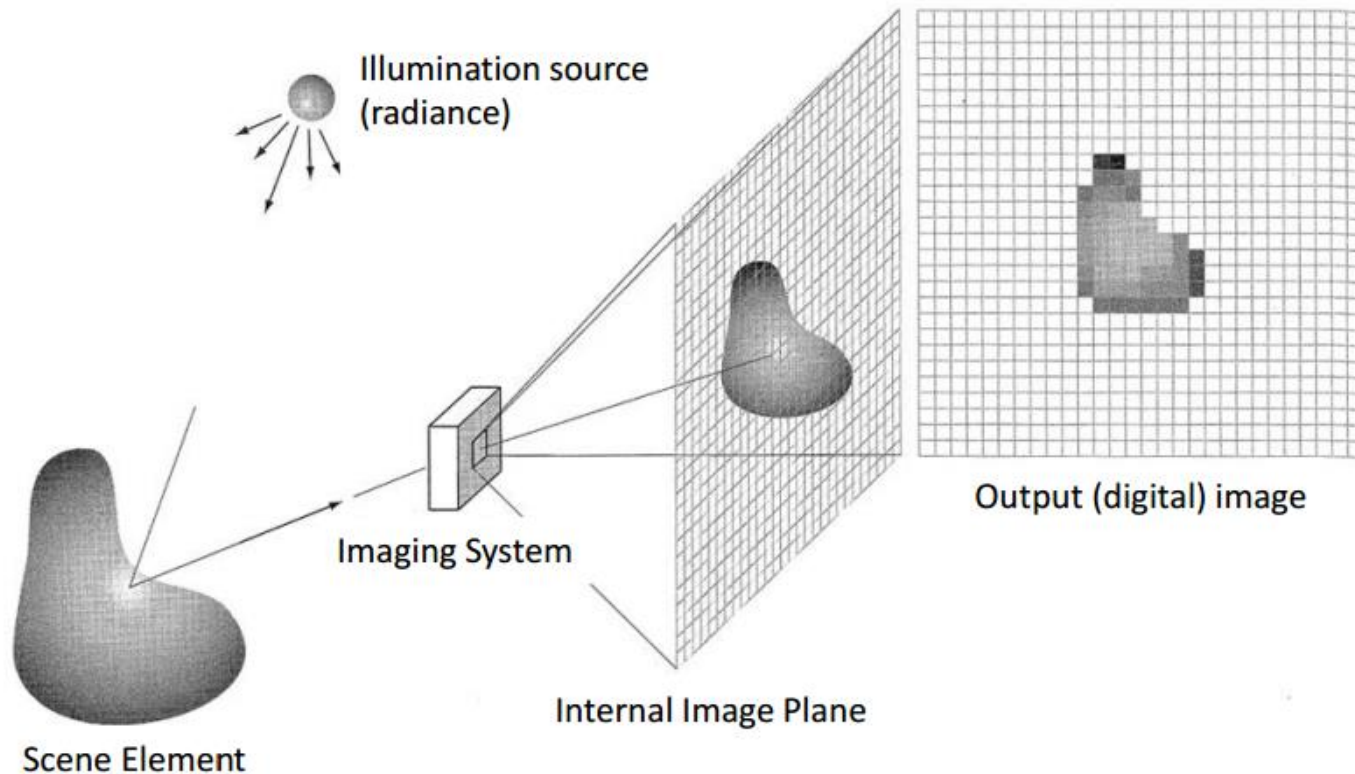
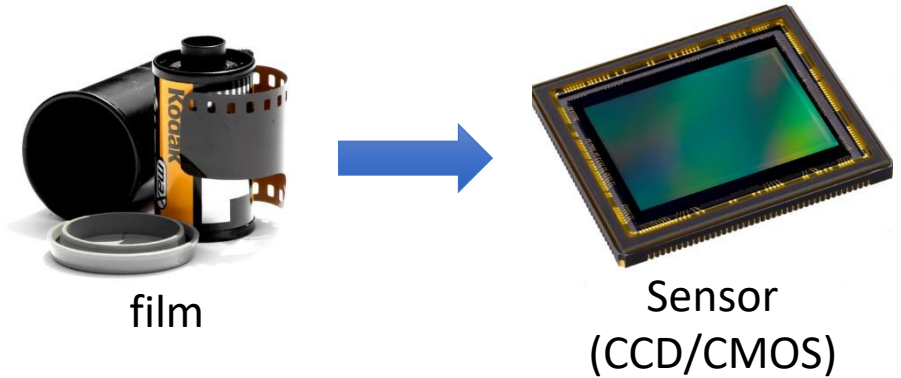


# Image Formation

- Shutter speed (exposure time) also matters

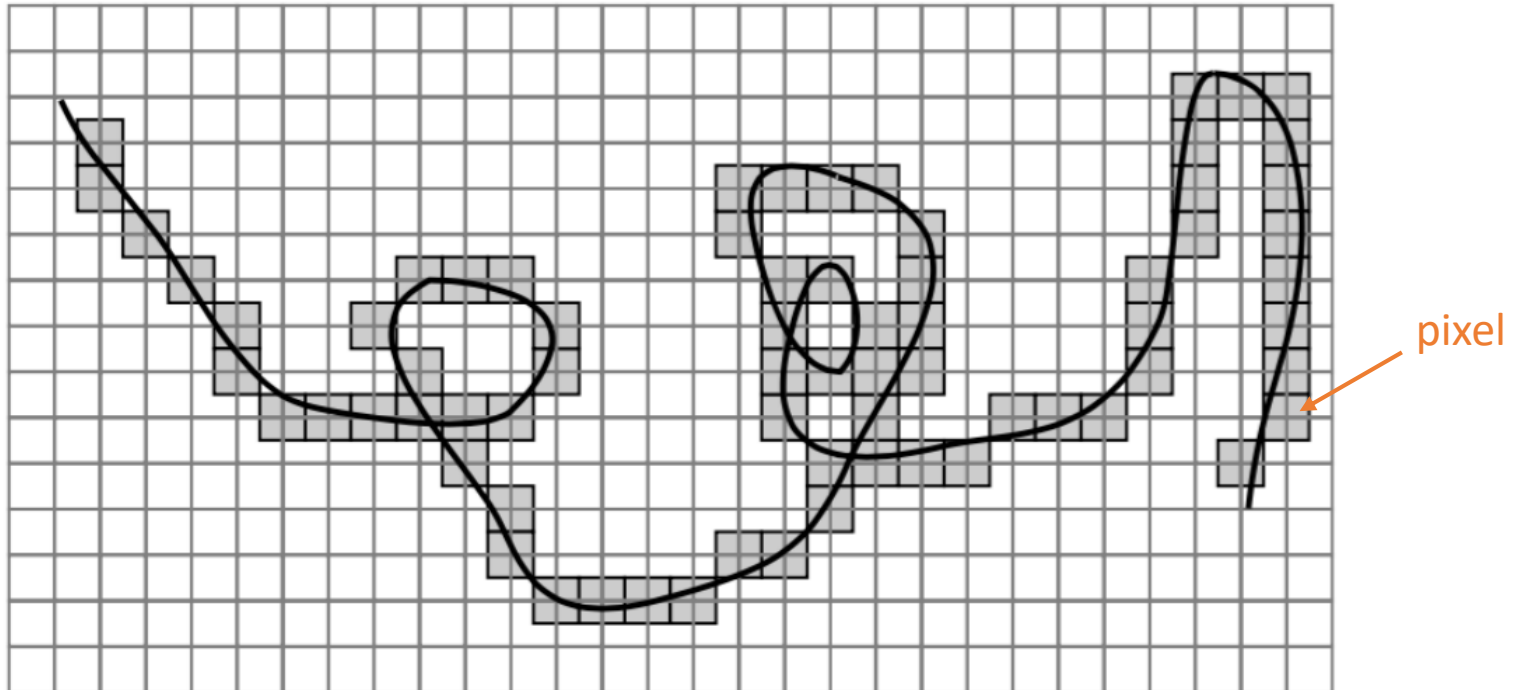


# Digital Imaging



# Digital Imaging

- Images are **sampled** and **quantized**
  - **Sampled:** discrete space (and time)
  - **Quantized:** only a finite number of possible values (*i.e.* 0 to 255)



Source: Ulas Bagci



# Digital Imaging

- Camera pipeline

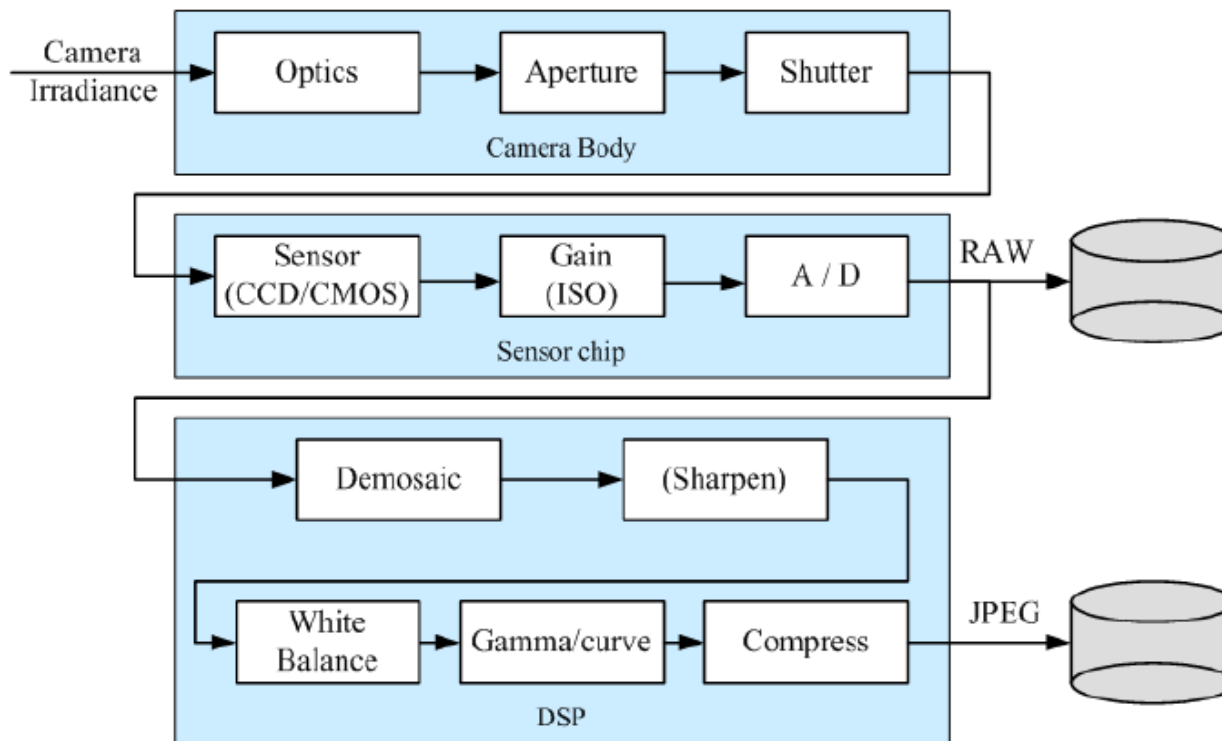
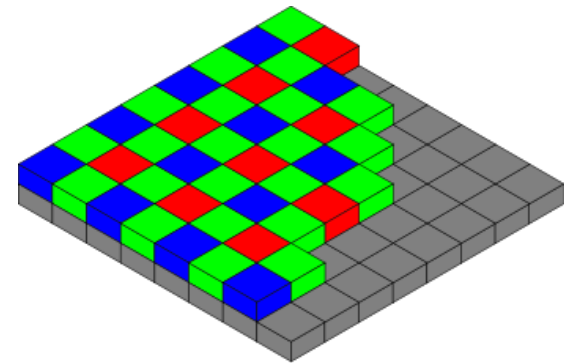
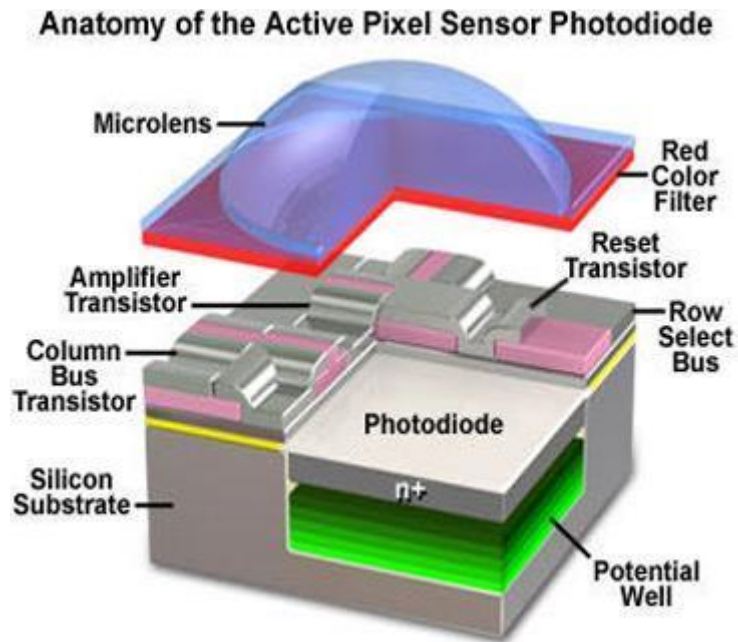


Figure 2.23 from *Computer Vision: Algorithms and Applications*

# Digital Imaging

- Demosaicing: color filter array interpolation
  - The image sensor knows nothing about color!



Color filter array (CFA)  
Bayer pattern: 1R1B2G in a 2x2 block

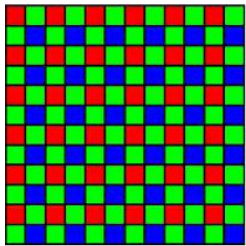
# Digital Imaging



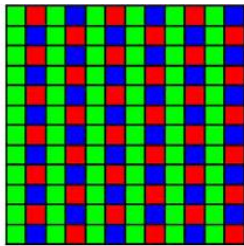
A picture of Alim Khan (1880-1944), Emir of Bukhara, taken in 1911.

# Digital Imaging

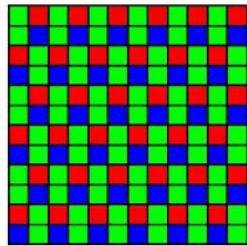
- More CFA design



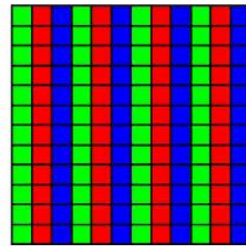
(a) Bayer [11]



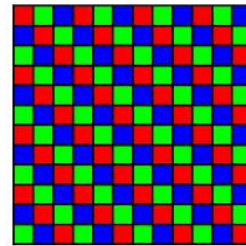
(b) Yamanaka [12]



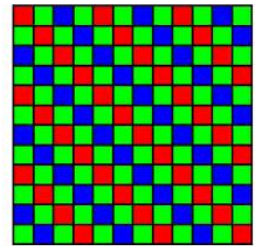
(c) Lukac [20]



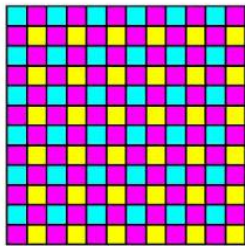
(d) Vertical



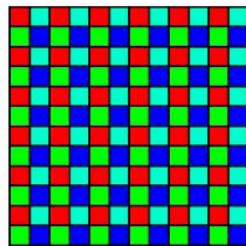
(e) Diagonal



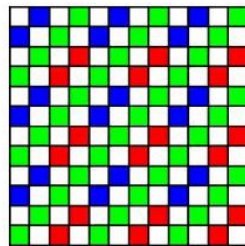
(f) Modified Bayer [20]



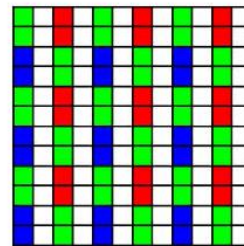
(g) Cyan-Magenta-Yell.



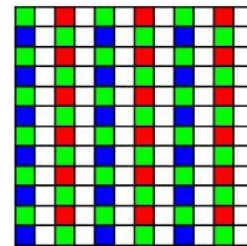
(h) Sony 4-Color [18]



(i) Kodak Ver. 1 [22]



(j) Kodak Ver. 2 [22]



(k) Kodak Ver. 3 [22]



# Digital Imaging

- Resolution
  - Image sensor samples and quantizes the scene

Low sampling rate may cause **aliasing** artifact



High resolution



Low resolution



Figure by Yen-Cheng Liu

# Digital Imaging

- Super resolution: the problem of resolving the high resolution image from the low resolution image

bicubic  
(21.59dB/0.6423)



SRResNet  
(23.53dB/0.7832)



SRGAN  
(21.15dB/0.6868)



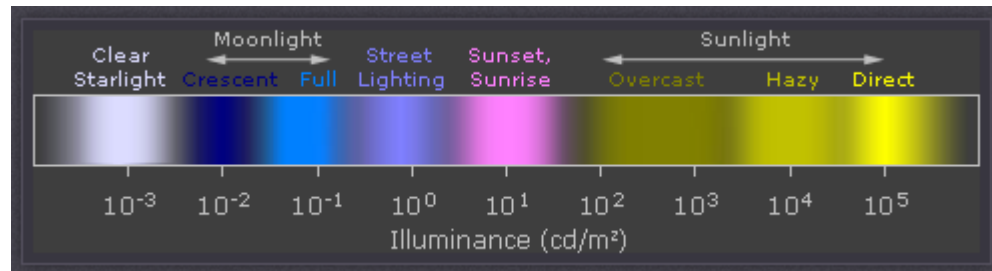
original



Example results of 4x upscaling. Figure from SRGAN [Ledig et al. CVPR 2017]

# Digital Imaging

- Dynamic range
  - Information loss due to A/D conversion
  - Typical image: 8 bit (0~255)



The world is HDR and our eyes have great ability to sense it



An exposure bracketed sequence (Each picture is a LDR image)



# Digital Imaging

- HDR imaging: LDRs  $\rightarrow$  HDR
- Tone mapping: HDR  $\rightarrow$  LDR
- Do we really need HDR?
  - Exposure fusion: LDRs  $\rightarrow$  LDR  
[Mertens et al. PG 2007]

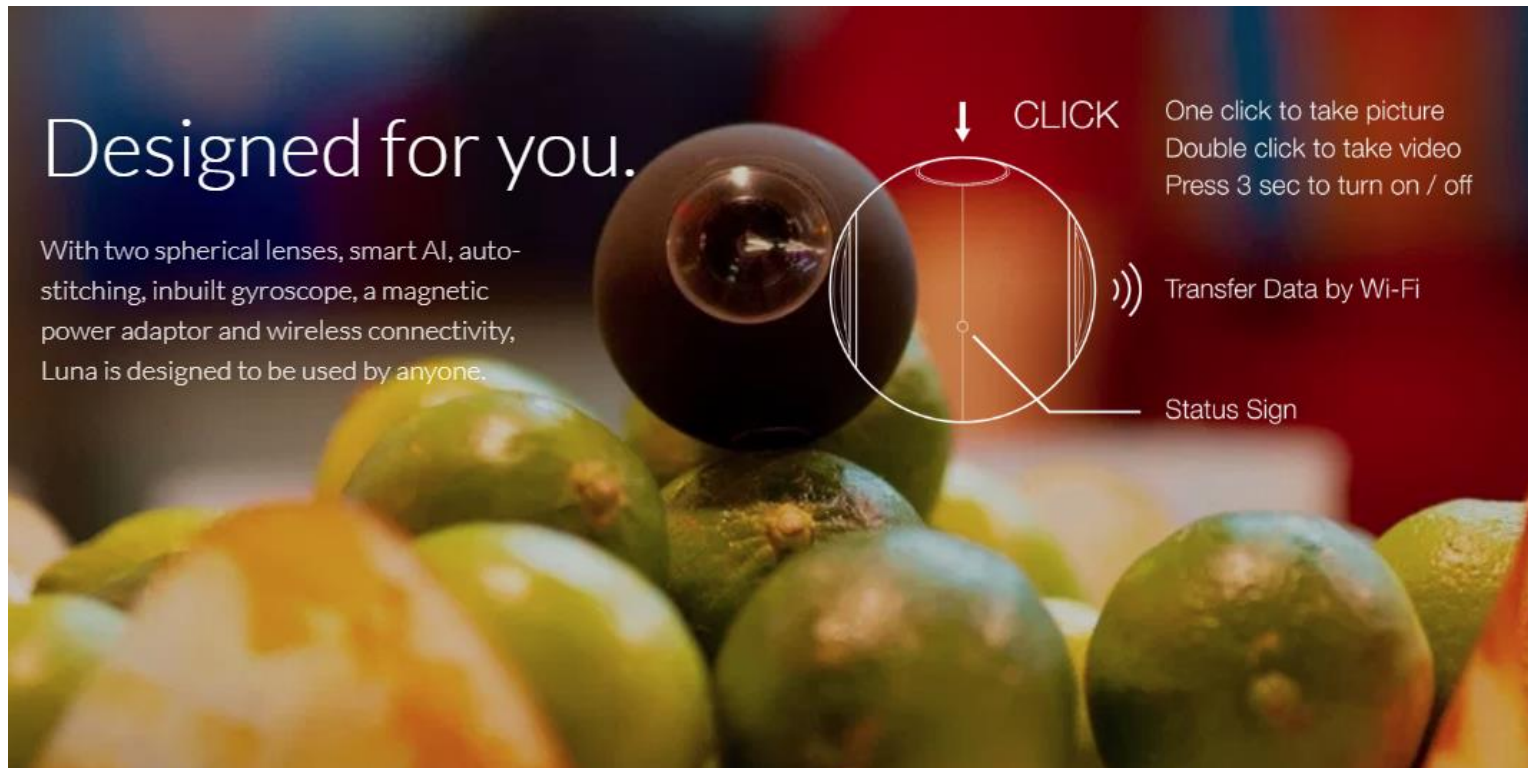


3 exposure (-2,0,+2) tone mapped image of a scene at Nippori Station.



# More Sensing Devices

- 360 camera



Source: LUNA

# More Sensing Devices

- Infra-red camera



# More Sensing Devices

- Depth camera



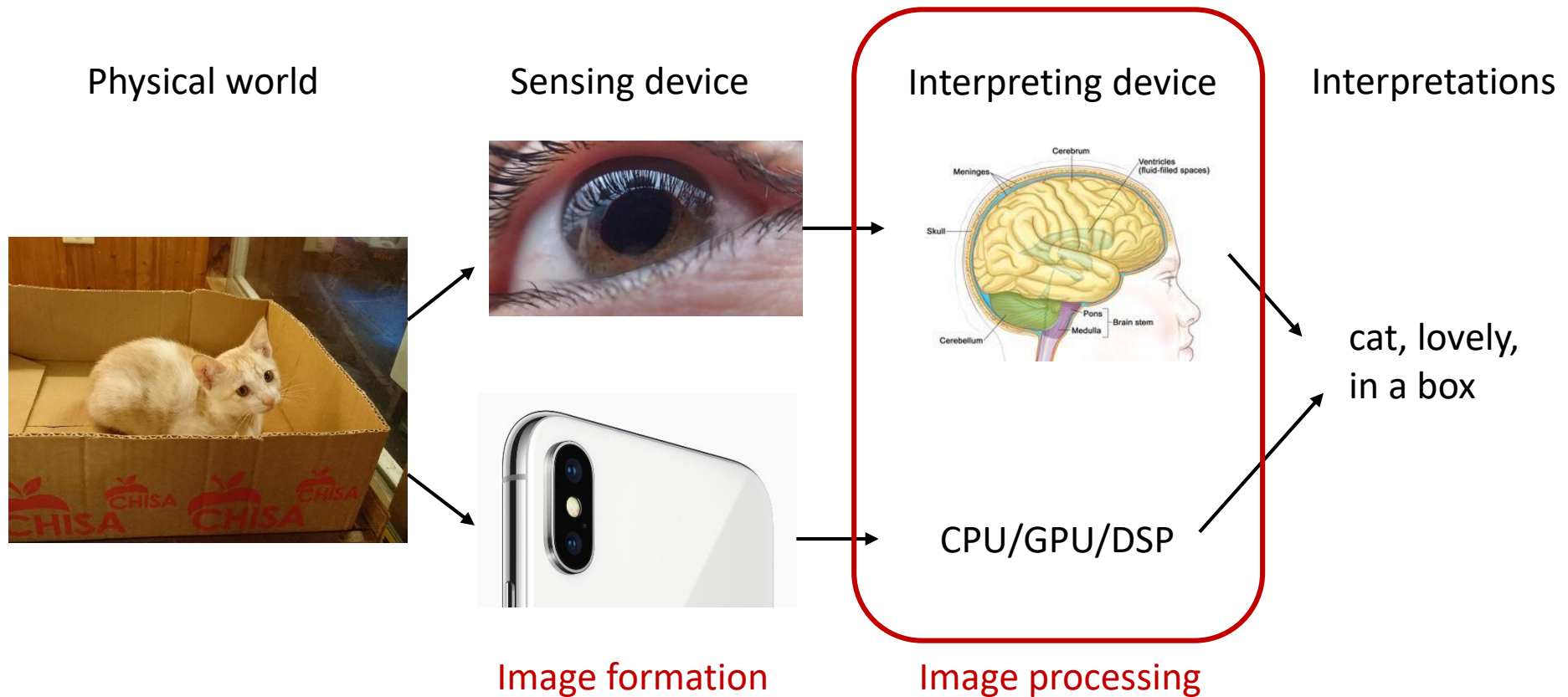
Kinect V2 (time of flight)



PointGrey Bumblebee 2 (stereo)

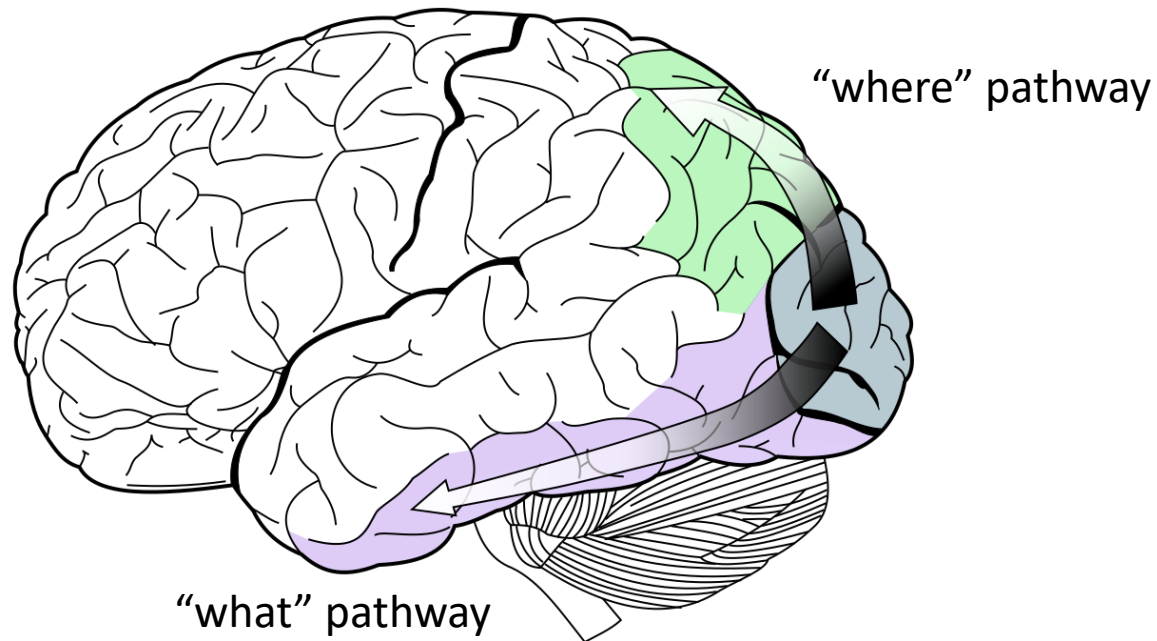
# Vision

- How vision is formed



# Image Processing in the Brain

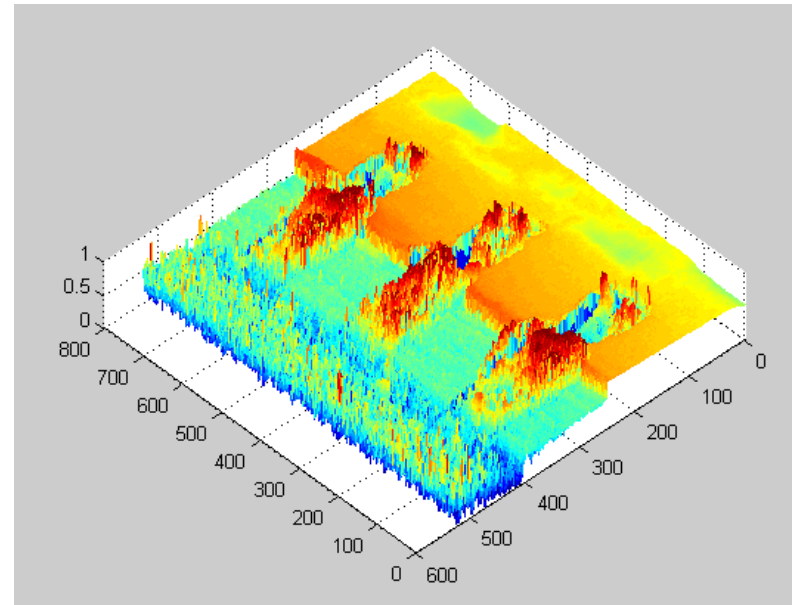
- The dorsal stream (green) and ventral stream (purple) are shown. They originate from a common source in the visual cortex.





# Digital Image Processing

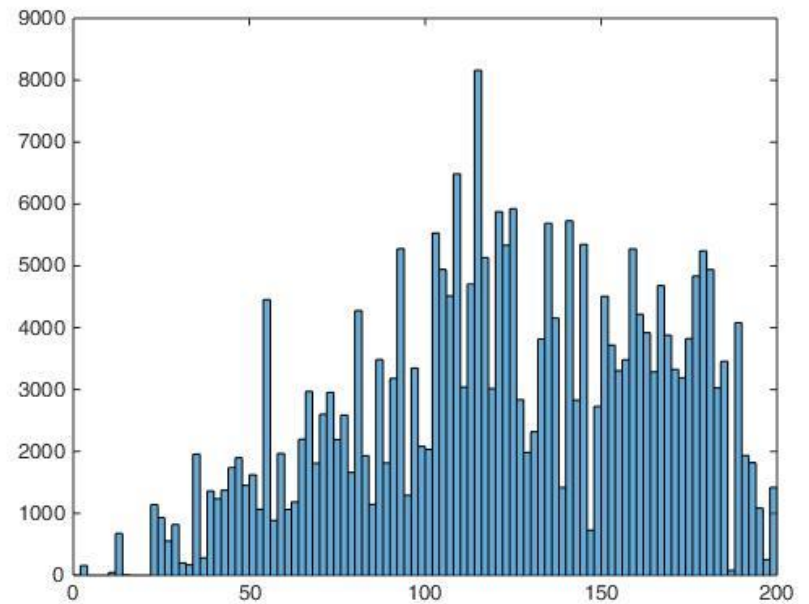
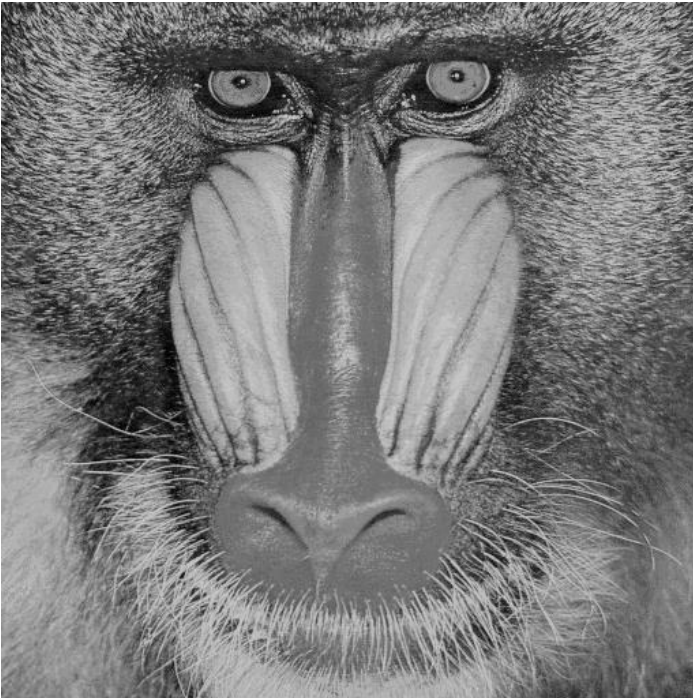
- Extract information (what and where) from digital images



# Digital Image Processing

- Some low-level image processing
  - Histogram
  - Morphological operations
  - Edge detection
  - Image filtering
- Topics to be covered in this course
  - Key point and feature descriptor
  - Matching
  - Geometric transformation
  - Semantic analysis
  - Learning-based techniques
  - ...

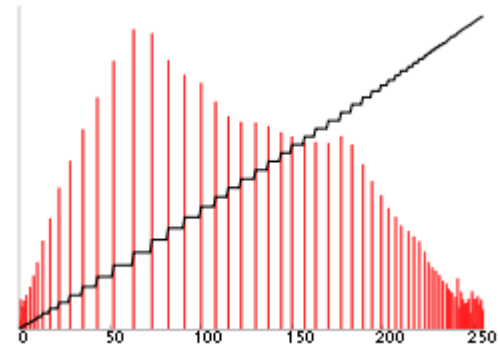
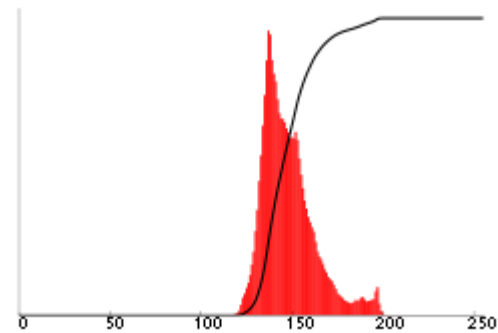
# Histogram





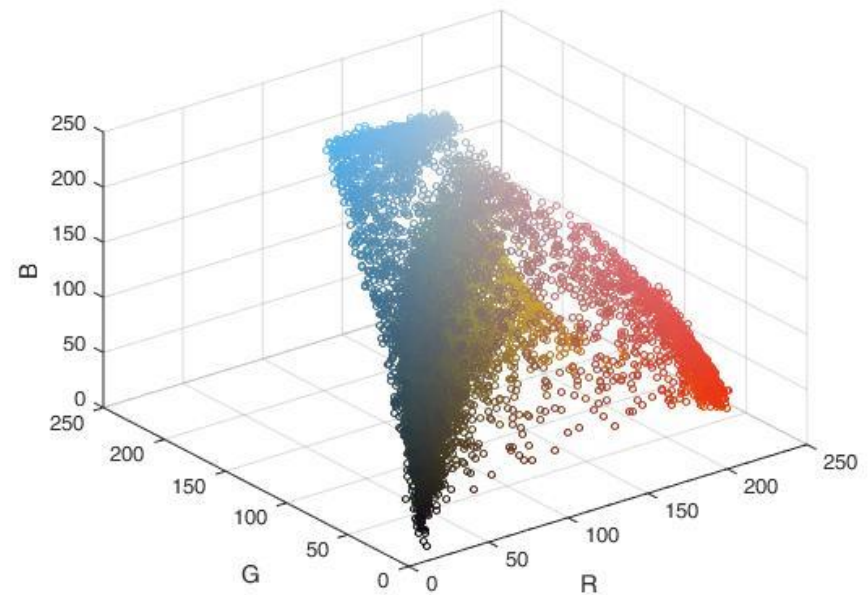
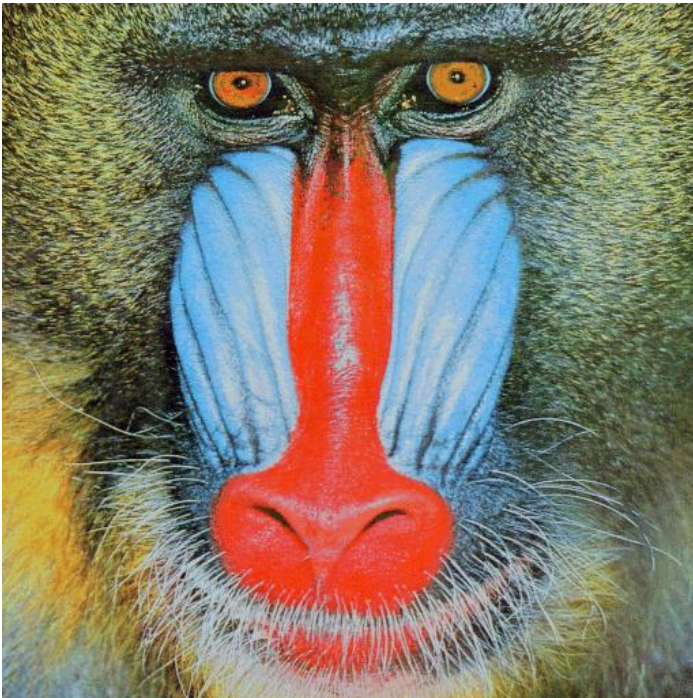
# Histogram

- Histogram equalization
  - By mapping CDF to the line  $y = x$



# Histogram

- Understanding data distribution



# Morphological Operations

- Take a **binary image** and a **structuring element** as input.



Dilation



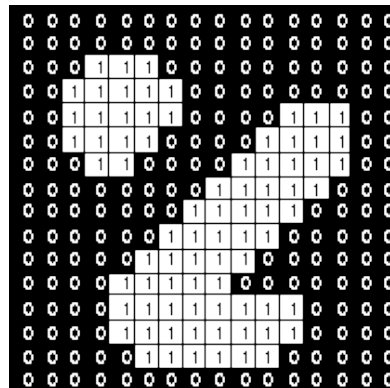
Erosion

1	1	1
1	1	1
1	1	1

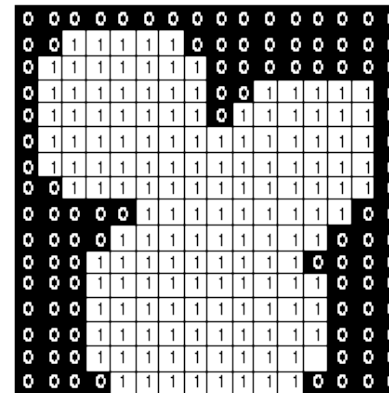
Example structuring element

Set of coordinate points =

{ (-1, -1), (0, -1), (1, -1),  
(-1, 0), (0, 0), (1, 0),  
(-1, 1), (0, 1), (1, 1) }



*dilate*



# Morphological Operations

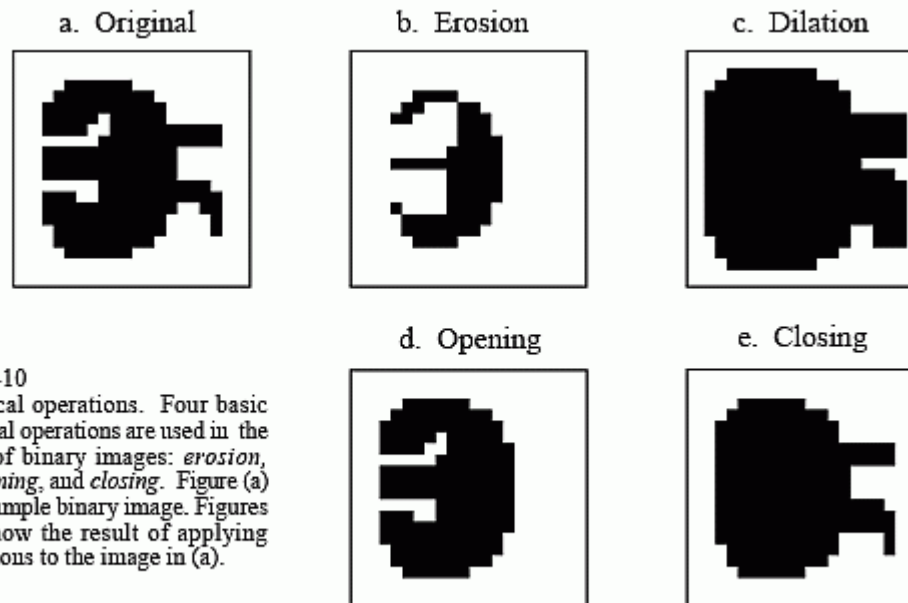


FIGURE 25-10  
Morphological operations. Four basic morphological operations are used in the processing of binary images: *erosion*, *dilation*, *opening*, and *closing*. Figure (a) shows an example binary image. Figures (b) to (e) show the result of applying these operations to the image in (a).

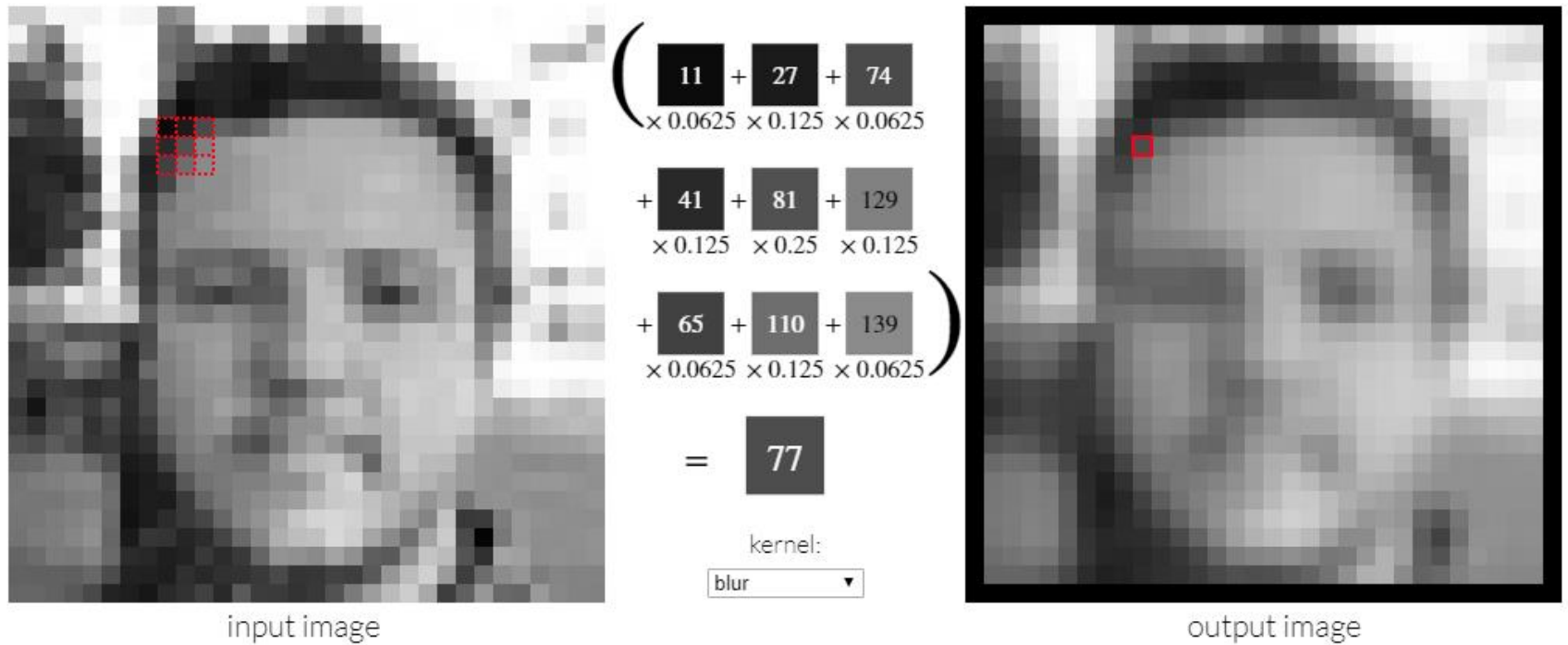
- Opening: erosion  $\rightarrow$  dilation
- Closing: dilation  $\rightarrow$  erosion

# Image Filtering

- What is filtering?
  - Forming a new image whose pixel values are transformed from original pixel values
- Goals
  - To extract useful information from images
    - *e.g.* edges
  - To transform images into another domain where we can modify/enhance image properties
    - *e.g.* denoising, image decomposition

# Image Filtering

- Try it yourself!



input image

$$\begin{aligned} & \left( \begin{array}{ccc} 11 & + & 27 & + & 74 \\ \times 0.0625 & \times 0.125 & \times 0.0625 \end{array} \right. \\ & + \begin{array}{ccc} 41 & + & 81 & + & 129 \\ \times 0.125 & \times 0.25 & \times 0.125 \end{array} \\ & \left. + \begin{array}{ccc} 65 & + & 110 & + & 139 \\ \times 0.0625 & \times 0.125 & \times 0.0625 \end{array} \right) \\ & = 77 \end{aligned}$$

kernel:  
blur

output image

<http://setosa.io/ev/image-kernels/>

# Image Filtering

- Convolution
  - Linear shift invariant (LSI)

$$g(x, y) = \frac{1}{W} \sum_{i, j \in [-r, r]} h(i, j) f(x - i, y - j)$$

$$W = \sum_{i, j \in [-r, r]} h(i, j)$$

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

$f(x, y)$

\*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

$h(x, y)$

=

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

$g(x, y)$

Output size changed... ☹️

Figure 3.10 from *Computer Vision: Algorithms and Applications*

# Image Filtering

- Padding



Zero padding



Symmetric



Replicate



Circular



# Image Filtering

- Box filter
  - Average filter
  - Compute summation if ignoring  $1/N$
  - Complexity:  $O(r^2)$

$$g(x, y) = \frac{1}{N} \sum_{i, j \in [-r, r]} f(x - i, y - j)$$

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

$f(x, y)$

*box filter*

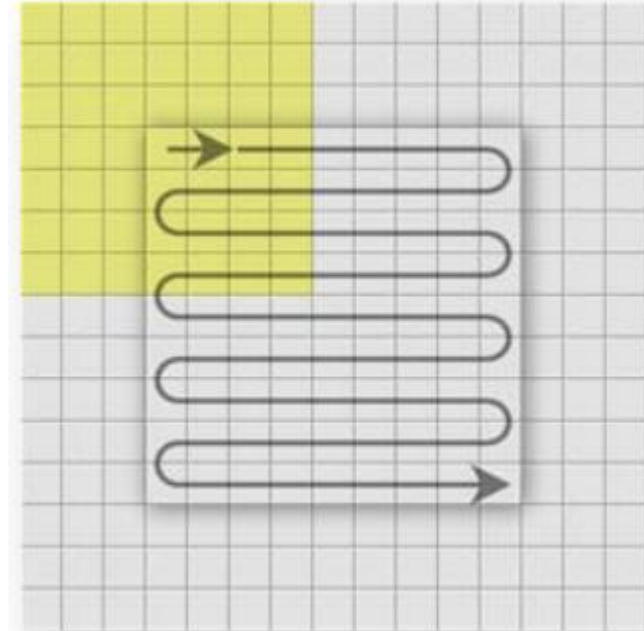
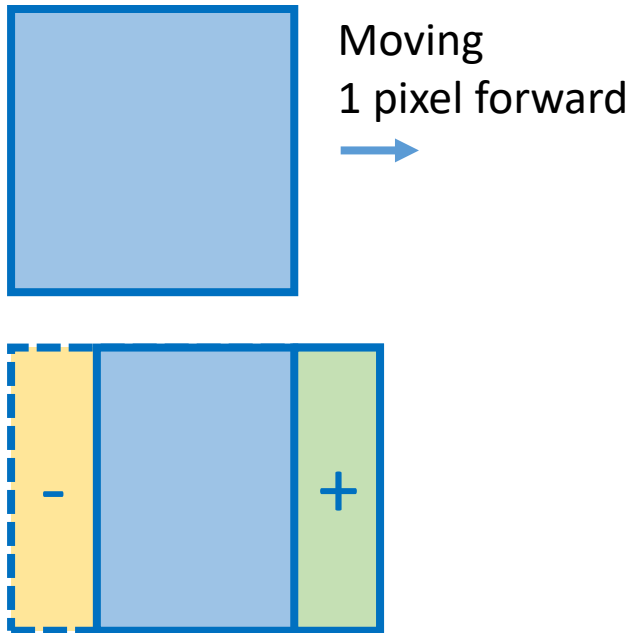


		$\frac{825}{9}$					

$g(x, y)$

# Image Filtering

- Box filter in  $O(r)$ 
  - Moving sum technique



Source: Ben Weiss

Note:  $O(1)$  filter is also called **constant time** filter

# Image Filtering

- Box filter in  $O(1)$ 
  - Integral image (sum area table)
    - Computing integral image: 2 addition + 1 subtraction
    - Obtaining box sum: 2 subtraction + 1 addition
    - Regardless of box size 😊

3	2	7	2	3
1	5	1	3	4
5	1	3	5	1
4	3	2	1	6
2	4	1	4	8

Sum = 24

3	5	12	14	17
4	11	19	24	31
9	17	28	38	46
13	24	37	48	62
15	30	44	59	81

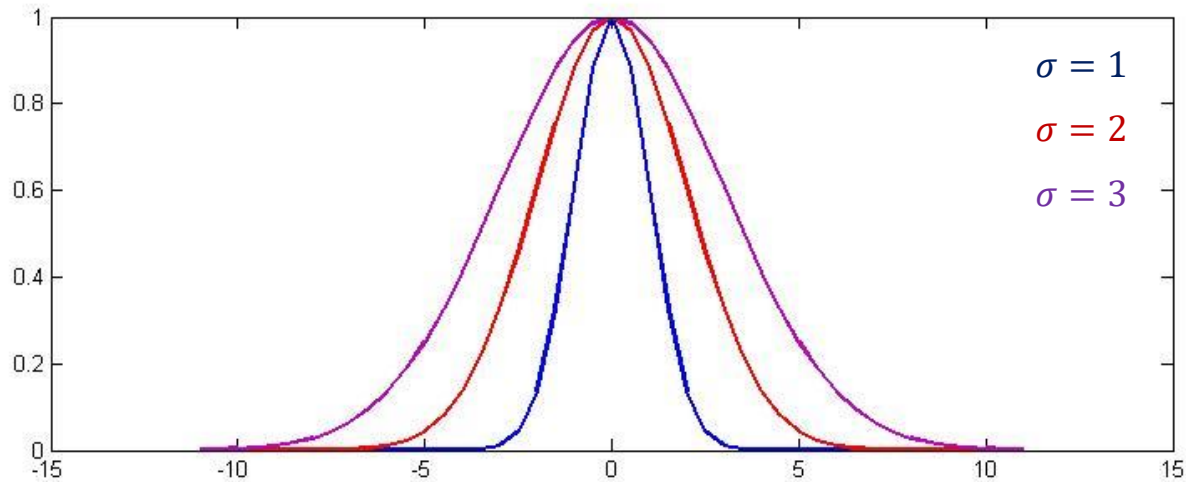
$19 + 17 - 11 + 3 = 28$

3	5	12	14	17
4	11	19	24	31
9	17	28	38	46
13	24	37	48	62
15	30	44	59	81

Sum =  $48 - 14 - 13 + 3 = 24$

# Image Filtering

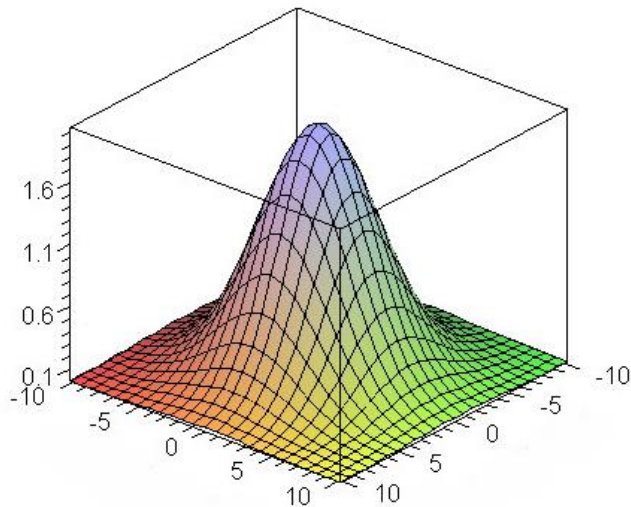
- Gaussian filter
  - The kernel weight is a Gaussian function  $h(x) = e^{-\frac{x^2}{2\sigma^2}}$
  - Center pixels contribute more weights



1D illustration of Gaussian functions

# Image Filtering

- Gaussian filter
  - 2D case:  $h(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$
  - Complexity:  $O(r^2)$



Original Image



Gaussian filtered image,  $\sigma = 2$



# Image Filtering

$$g(x, y) = \frac{1}{W} \sum_{i, j \in [-r, r]} h(i, j) f(x - i, y - j)$$

$$W = \sum_{i, j \in [-r, r]} h(i, j)$$

- Gaussian filter in  $O(r)$ 
  - Gaussian kernel is **separable**  
(The same technique can be applied to other separable kernels)

$$h(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} = e^{-\frac{x^2}{2\sigma^2}} \cdot e^{-\frac{y^2}{2\sigma^2}}$$

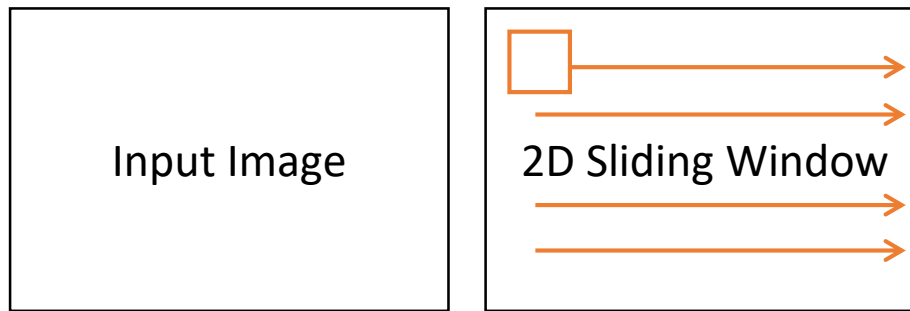
$$g(x, y) = \frac{1}{W} \sum_{i \in [-r, r]} \sum_{j \in [-r, r]} e^{-\frac{x^2 + y^2}{2\sigma^2}} f(x - i, y - j)$$

$$g(x, y) = \frac{1}{W} \sum_{j \in [-r, r]} e^{-\frac{y^2}{2\sigma^2}} \sum_{i \in [-r, r]} e^{-\frac{x^2}{2\sigma^2}} f(x - i, y - j)$$

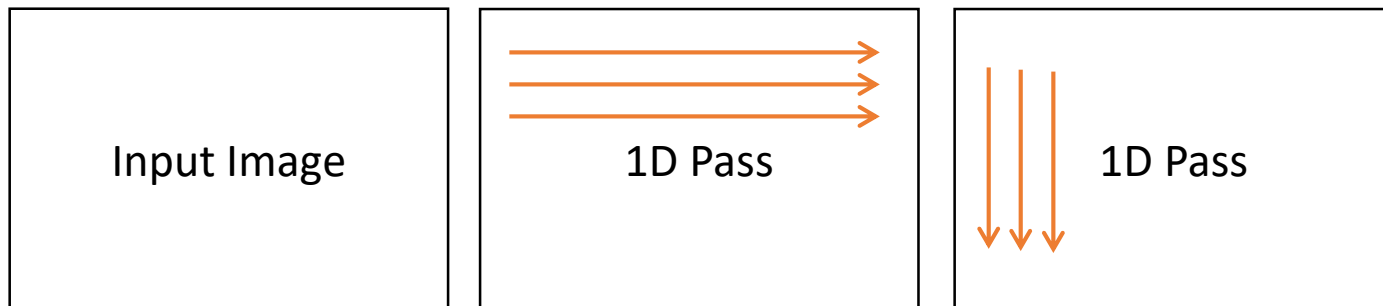
# Image Filtering

- Gaussian filter in  $O(r)$

Direct 2D implementation:  $O(r^2)$



Separable implementation:  $O(r)$



# Image Filtering

- Gaussian filter in  $O(1)$ 
  - FFT
  - Iterative box filtering
  - Recursive filter



# Image Filtering

- $O(1)$  Gaussian filter by FFT approach

$$g = h * f$$

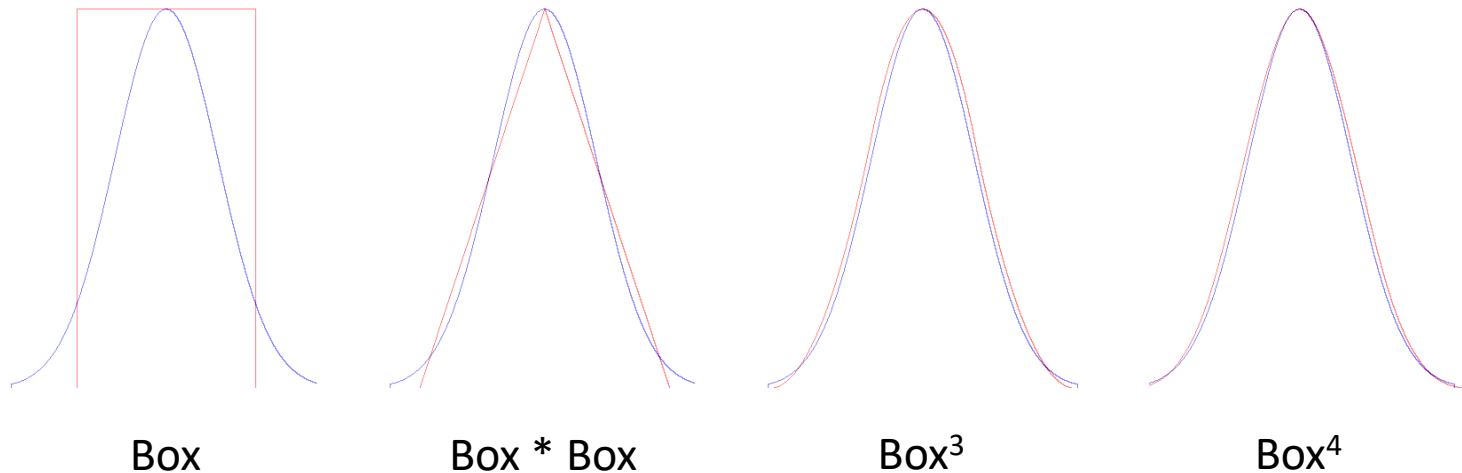
$$\mathcal{F}(g) = \mathcal{F}(h) \cdot \mathcal{F}(f)$$

$$g = \mathcal{F}^{-1}(\mathcal{F}(h) \cdot \mathcal{F}(f))$$

- Complexity:
  - Take FFT:  $O(wh \ln(w) \ln(h))$
  - Multiply by FFT of Gaussian:  $O(wh)$
  - Take inverse FFT:  $O(wh \ln(w) \ln(h))$
  - Cost independent of filter size

# Image Filtering

- $O(1)$  Gaussian filter by iterative box filtering
  - Based on the **central limit theorem**
  - Pros: easy to implement!
  - Cons: limited choice of box size (3, 5, 7, ...) results in limited choice of Gaussian function  $\sigma^2$



# Image Filtering

- $O(1)$  Gaussian filter by recursive implementation
  - All filters we discussed above are FIR filters
  - We can use IIR (infinite impulse response) filters to approximate Gaussians...

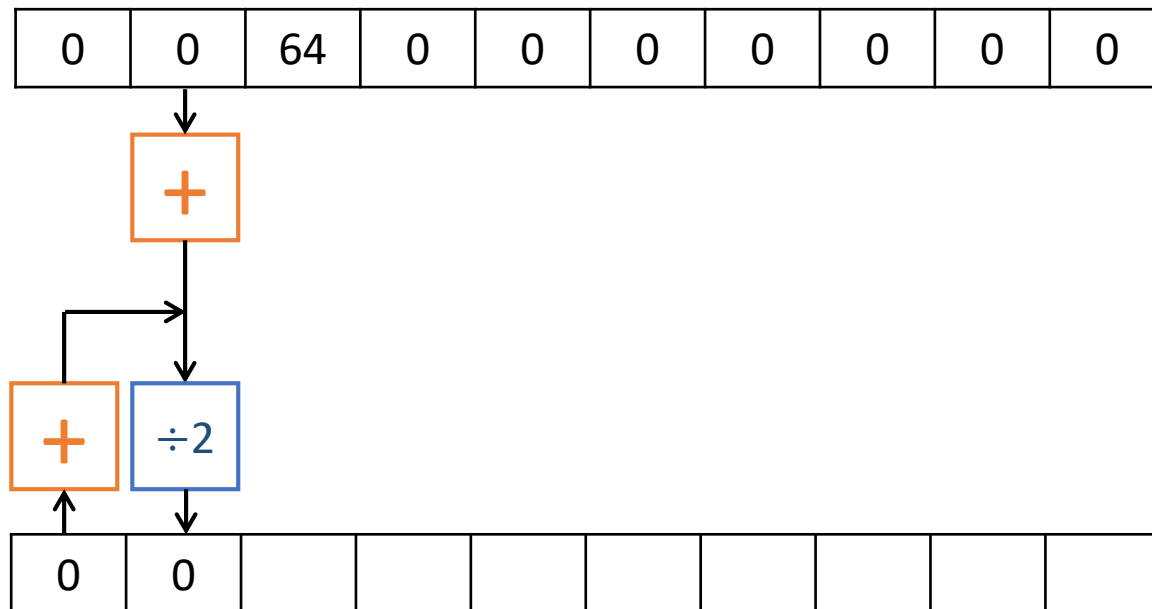
1<sup>st</sup> order IIR filter:

$$g(x) = a_0 \cdot f(x) - b_1 \cdot g(x - 1)$$

2<sup>nd</sup> order IIR filter:

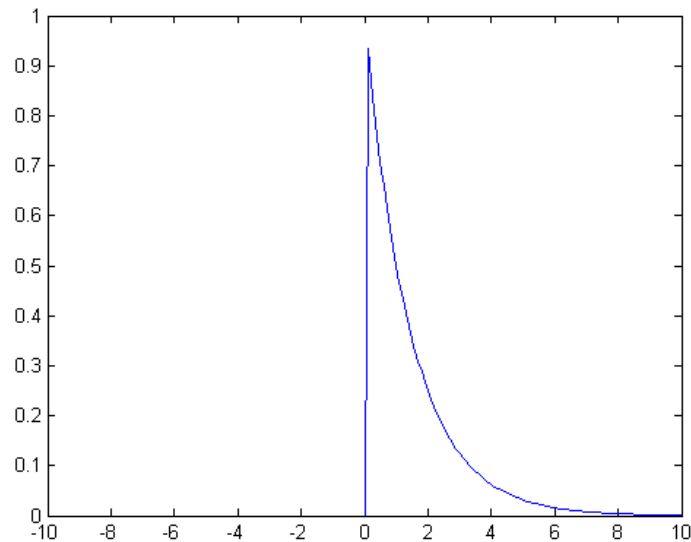
$$g(x) = a_0 \cdot f(x) + a_1 \cdot f(x - 1) - b_1 \cdot g(x - 1) - b_2 \cdot g(x - 2)$$

# IIR Filters



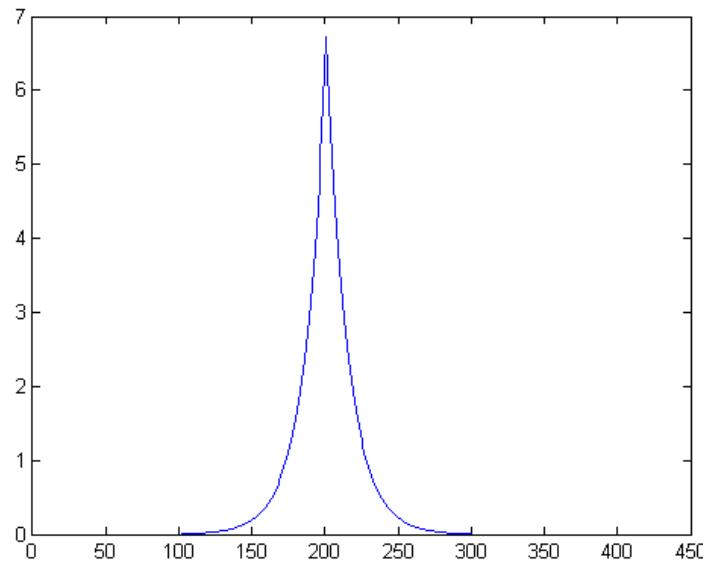
# IIR Filters

- The example above is an exponential decay
- Equivalent to convolution by:



# IIR Filters

- Makes large, smooth filters with very little computation! 😊
- One forward pass (causal), one backward pass (anti-causal), equivalent to convolution by:



# Image Filtering

- $O(1)$  Gaussian filter by recursive implementation
  - 2<sup>nd</sup> order IIR filter approximation

$$g(x) = a_0 \cdot f(x) + a_1 \cdot f(x - 1) - b_1 \cdot g(x - 1) - b_2 \cdot g(x - 2)$$

$$g'(x) = a_2 \cdot f(x + 1) + a_3 \cdot f(x + 2) - b_1 \cdot g'(x + 1) - b_2 \cdot g'(x + 2)$$

$$a_0 = (1 - e^{-\frac{1.695}{\sigma_S}})^2 / (1 + 3.39e^{-\frac{1.695}{\sigma_S}} / \sigma_S - e^{-\frac{3.39}{\sigma_S}})$$

$$a_1 = (1.695/\sigma_S - 1)e^{-\frac{1.695}{\sigma_S}} a_0,$$

$$b_1 = -2e^{-\frac{1.695}{\sigma_S}},$$

$$b_2 = e^{-\frac{3.39}{\sigma_S}}.$$

$$a_2 = (1.695/\sigma_S + 1)e^{-\frac{1.695}{\sigma_S}} a_0 \text{ and } a_3 = -a_0 b_2$$

“Recursively implementing the Gaussian and its derivatives”, ICIP 1992



# Image Filtering

- Median filter
  - 3x3 example:

11	19	22
12	25	27
18	26	23

A local patch

Sort

11, 12, 18, 19, 22, 23, 25, 26, 27

Replace center pixel value by median

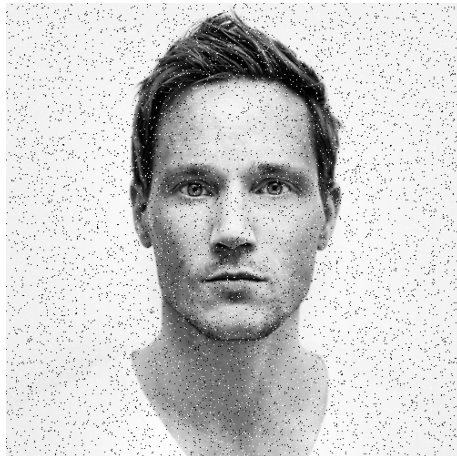
11	19	22
12	22	27
18	26	23

# Image Filtering

- Median filter
  - Useful to deal with **salt and pepper noise**



Original



Add salt & pepper

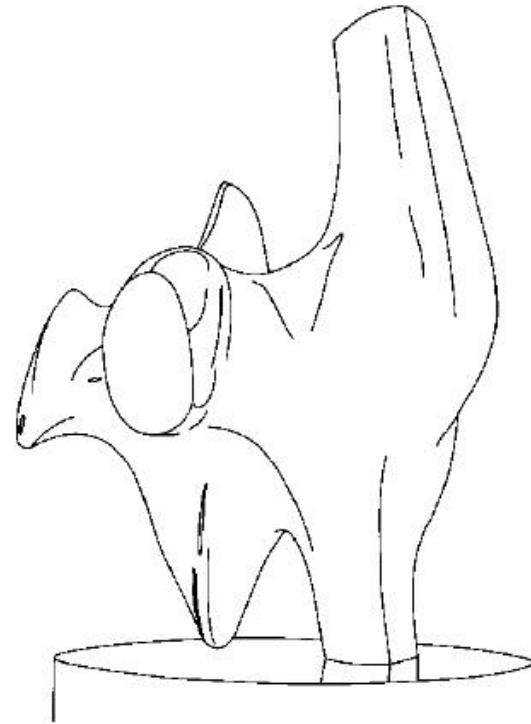
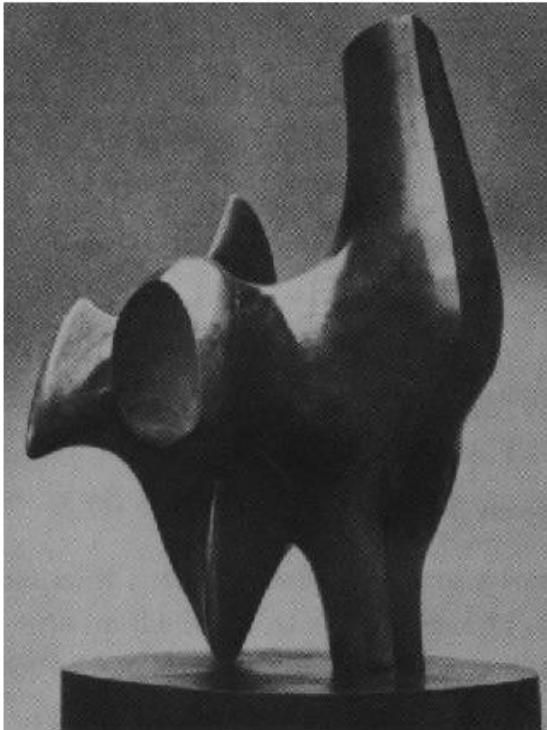


Gaussian filter



Median filter

# Edge Detection

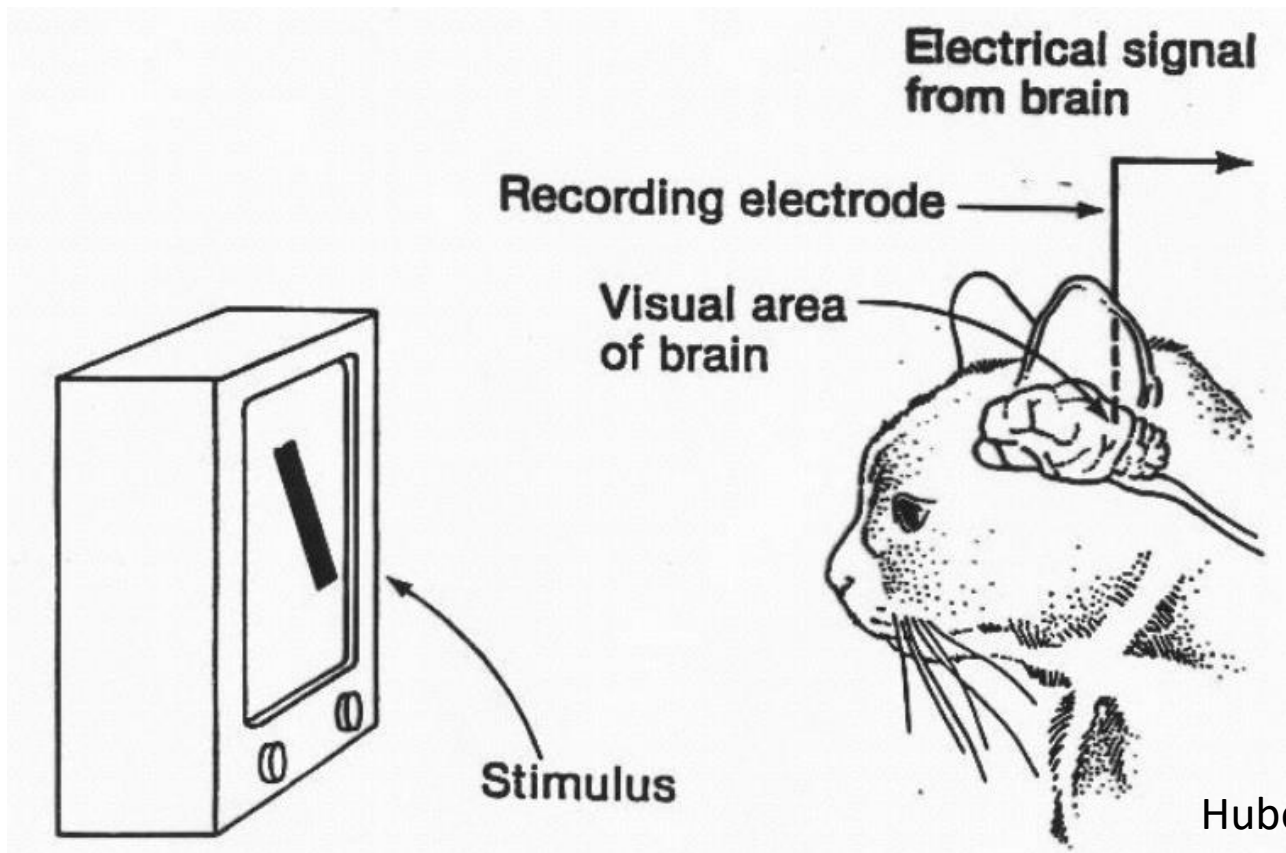


Convert a 2D image into a set of curves

- Extract salient features of the scene
- More compact than pixels

# Edge Detection

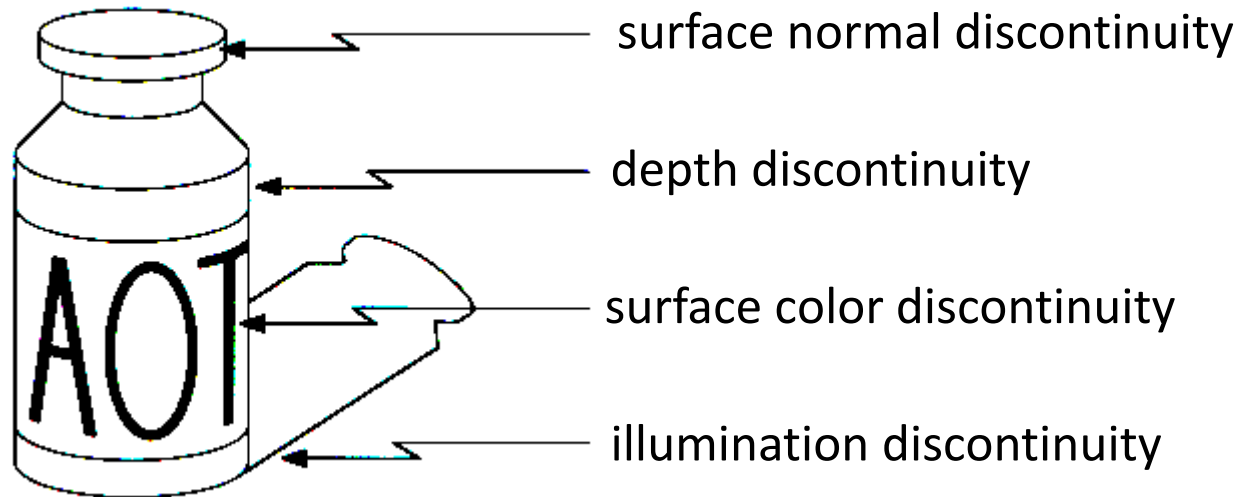
- We know edges are special from mammalian vision studies



Hubel & Wiesel (1962)

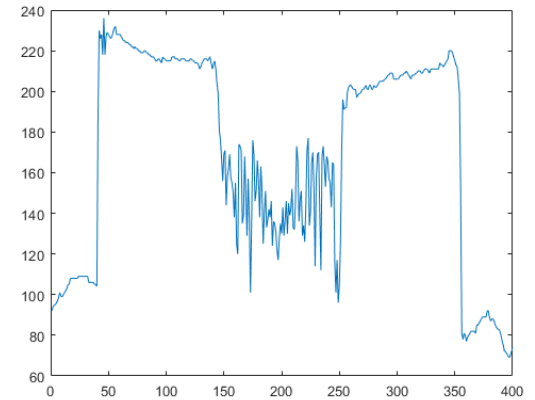
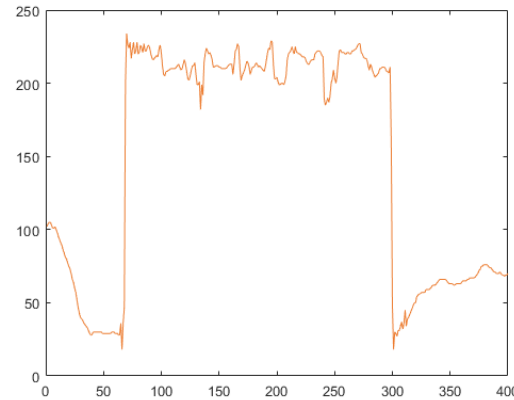
# Edge Detection

- Origin of edges

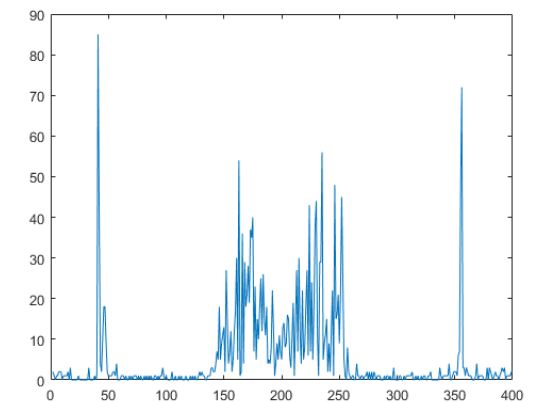
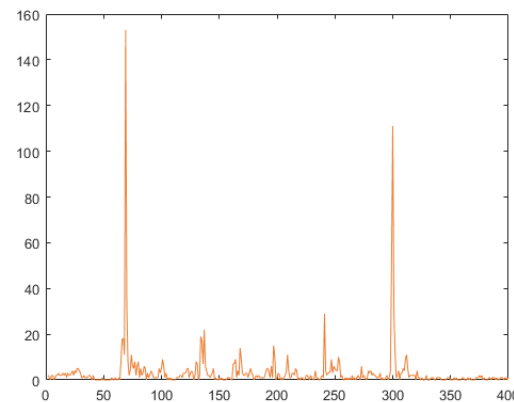


# Edge Detection

- Characterizing edges



Intensity profile



Gradient magnitude

# Edge Detection

- How to compute gradient for digital images?

$$\nabla f = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$

- Take **discrete derivative**

$$\frac{\partial f}{\partial x} \approx f(x+1, y) - f(x, y)$$

- Gradient direction and magnitude

$$\theta = \tan^{-1} \left( \frac{\partial f / \partial y}{\partial f / \partial x} \right) \quad \|\nabla f\| = \sqrt{\left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2}$$



# Discrete Derivative

- Backward difference

Equivalent to convolve with:

$$\frac{df}{dx} = f(x) - f(x - 1) \quad [1, -1]$$

- Forward difference

$$\frac{df}{dx} = f(x) - f(x + 1) \quad [-1, 1]$$

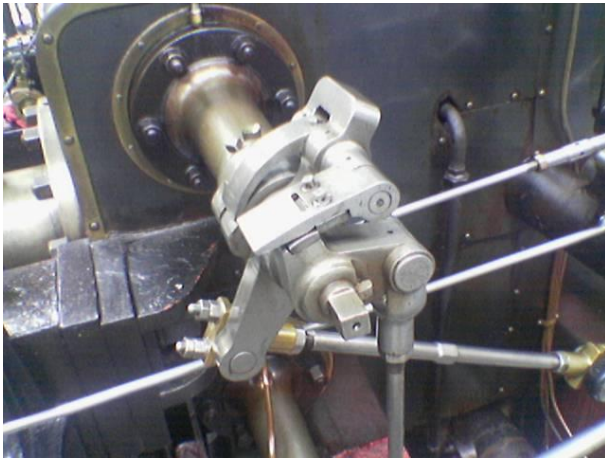
- Central difference

$$\frac{df}{dx} = f(x + 1) - f(x - 1) \quad [1, 0, -1]$$

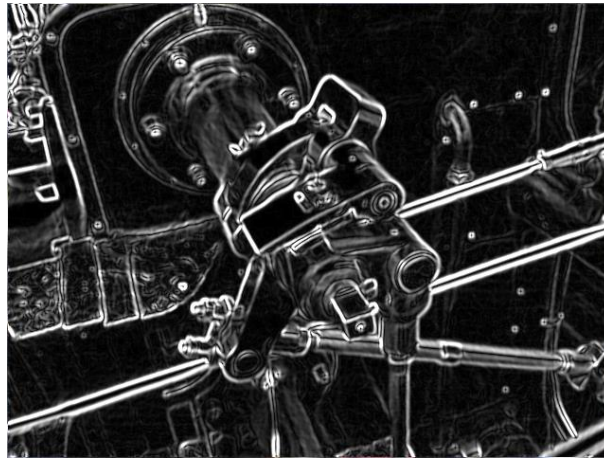
# Edge Detection

- Sobel filter

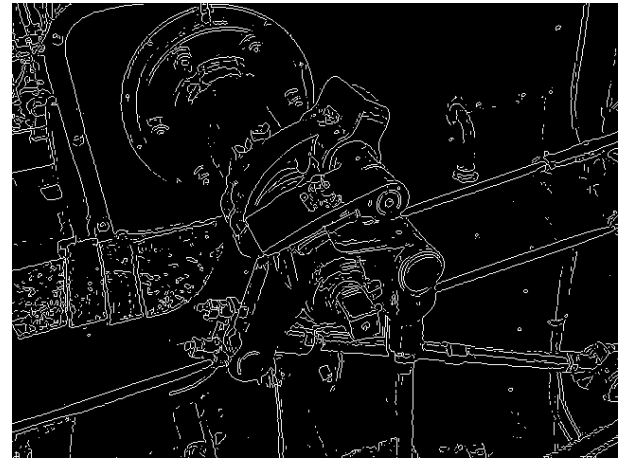
$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * f \quad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * f \quad G = \sqrt{G_x^2 + G_y^2}$$



Input image



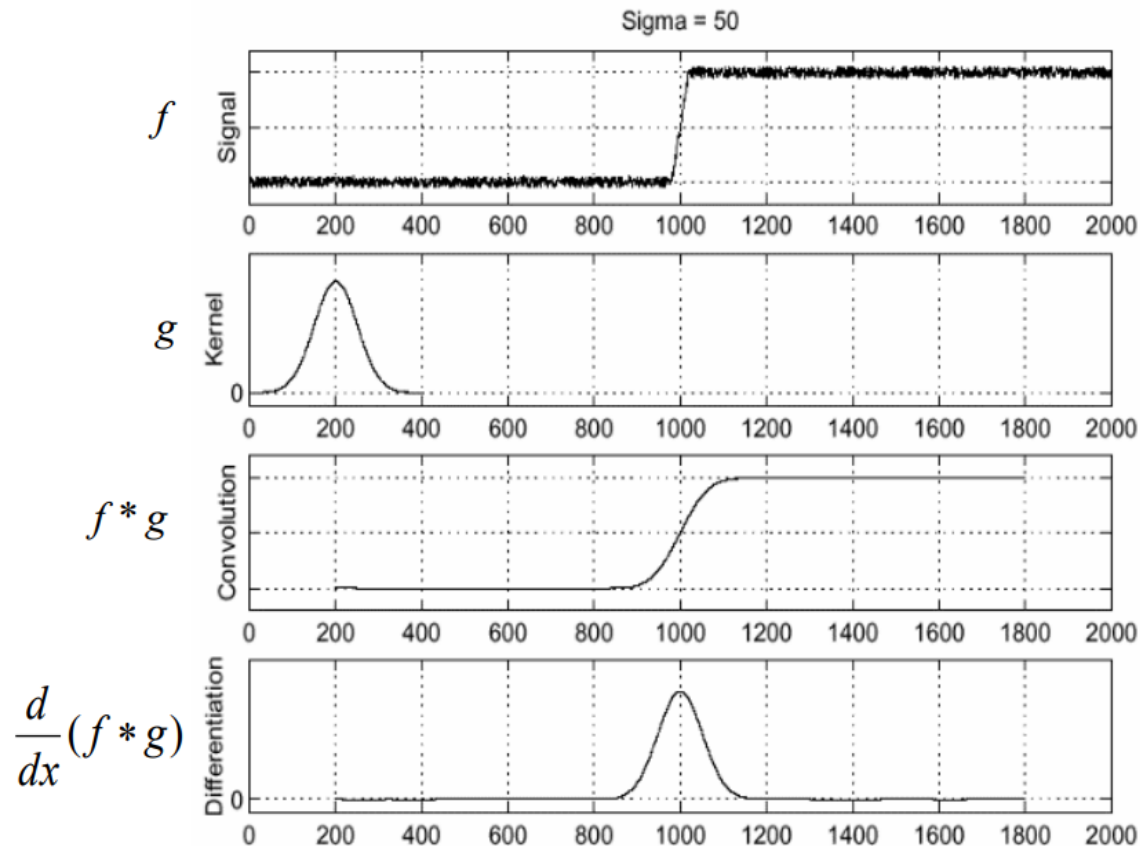
$G$



After thresholding

# Edge Detection

- Effect of noise
  - Difference filters respond strongly to noise
  - Solution: smooth first

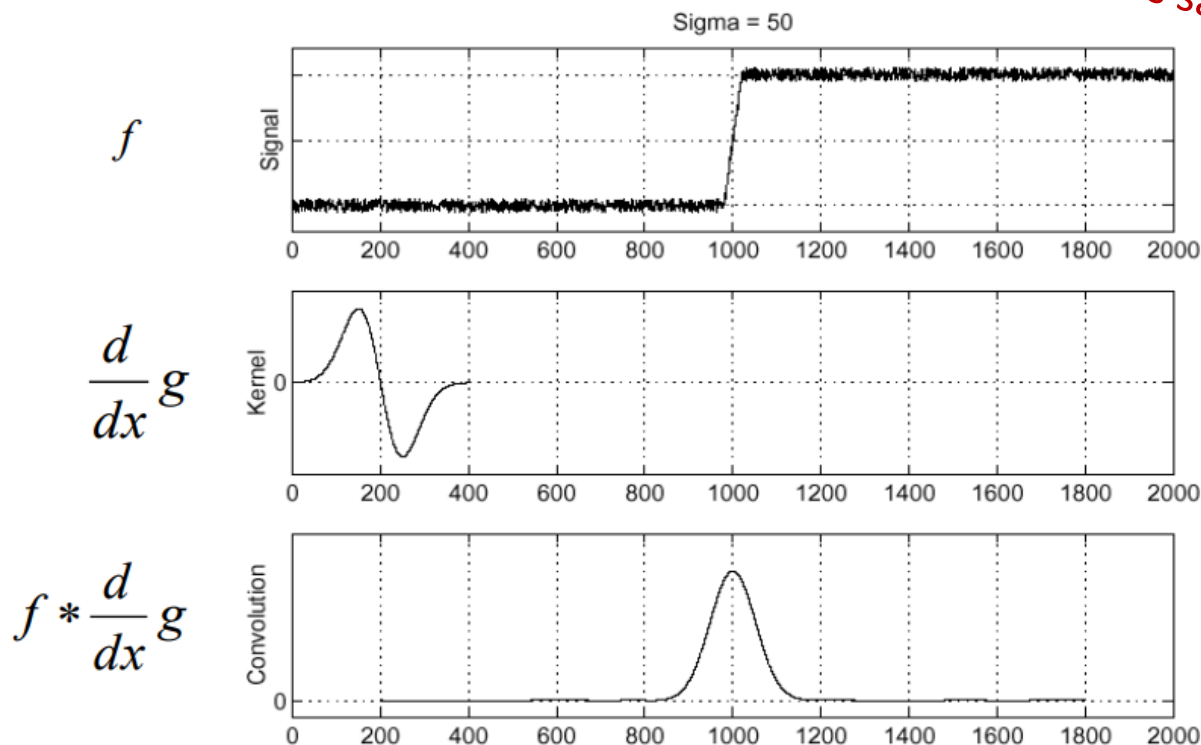


# Derivative Theorem of Convolution

- Differentiation is convolution, which is associative:

$$\frac{d}{dx}(f * g) = f * \left(\frac{d}{dx}g\right)$$

*This saves one operation!*

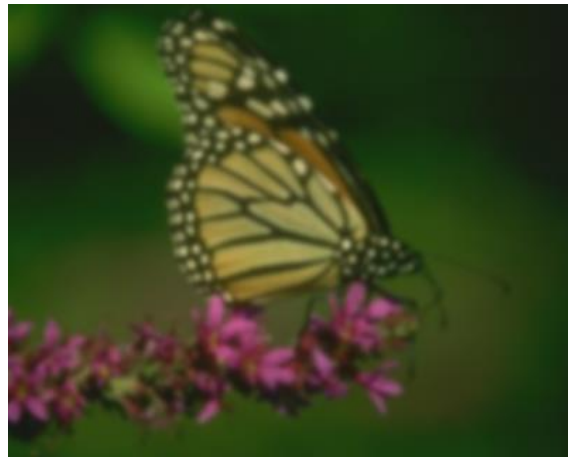


# Edge Detection

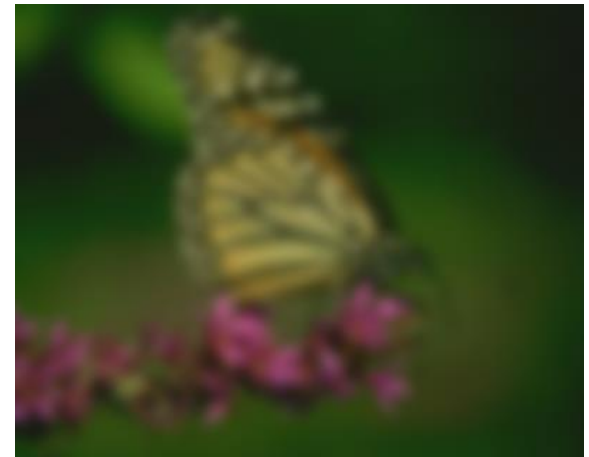
- Tradeoff between smoothing and localization
  - Smoothing filter removes noise but blurs edges ☹️



No filtering



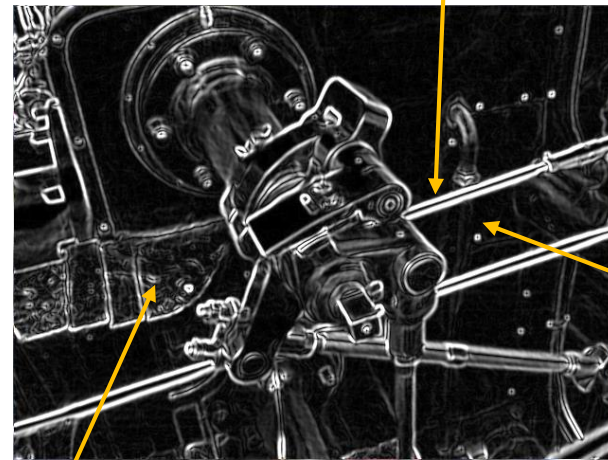
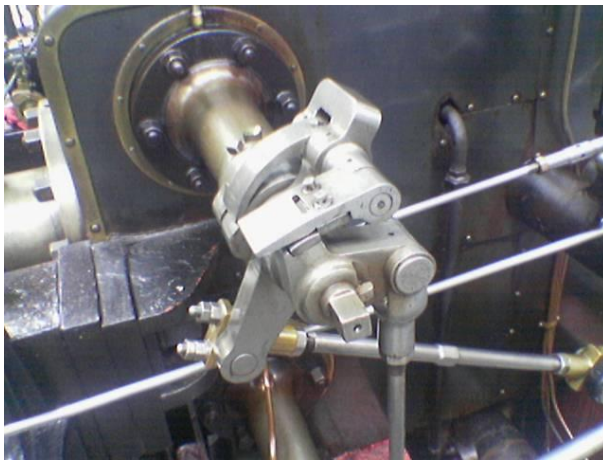
Gaussian filter,  $\sigma = 2$



Gaussian filter,  $\sigma = 5$

# Edge Detection

- Criteria for a good edge detector
  - Good detection
    - Find all real edges, ignoring noise
  - Good localization
    - Locate edges as close as possible to the true edges
    - Edge width is only one pixel



Bad localization

Missing edge

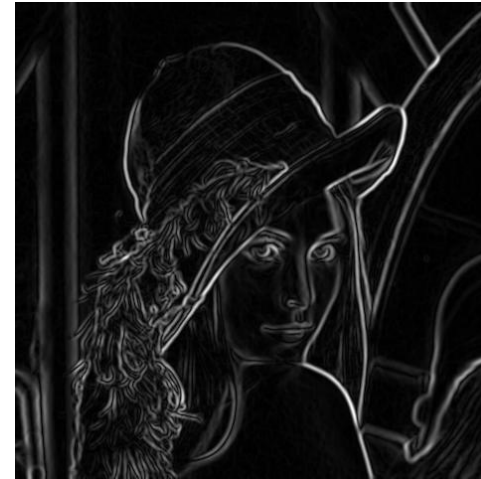
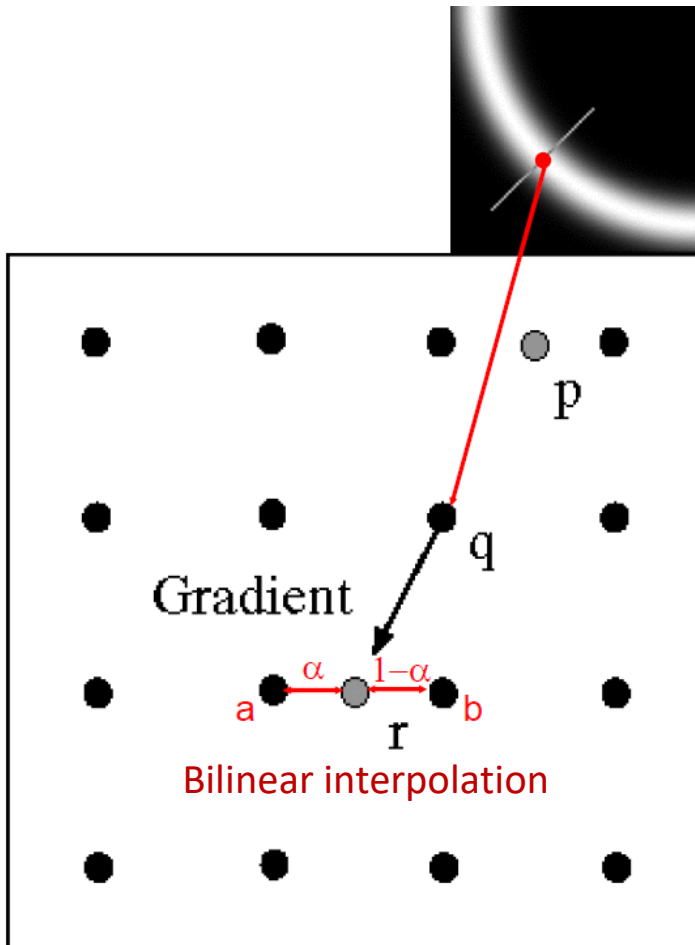
Not edge

# Edge Detection

- Canny edge detector
  - The most widely used edge detector
  - The best you can find in existing tools like MATLAB, OpenCV...
- Algorithm:
  - Apply Gaussian filter to reduce noise
  - Find the intensity gradients of the image
  - Apply **non-maximum suppression** to get rid of false edges
  - Apply **double threshold** to determine potential edges
  - Track edge by **hysteresis**: suppressing weak edges that are not connected to strong edges



# Non-Maximum Suppression



Gradient magnitude



After NMS

# Double Thresholding



High threshold



**Strong edges**

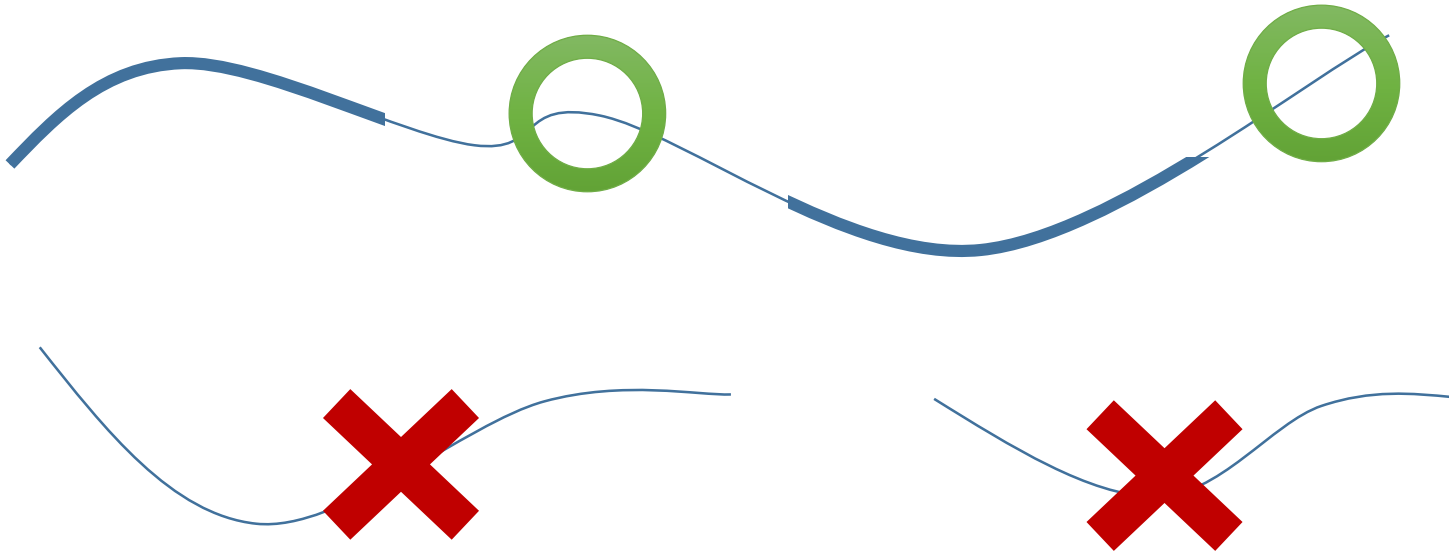
Low threshold



Weak edges

# Hysteresis

- Find **connected components** from strong edge pixels to finalize edge detection



# More Image Filtering

- Bilateral filter
  - Smoothing images while preserving edges



Input



Output

Small fluctuations are removed

Edge remains sharp

# Bilateral Filtering

- Bilateral filter

$$g(x, y) = \frac{1}{W} \sum_{i, j \in [-r, r]} h(i, j) f(x - i, y - j)$$

$$g(x, y) = \frac{1}{W} \sum_{i, j \in [-r, r]} \underbrace{h_s(i, j)}_{\text{Spatial kernel}} \cdot \underbrace{h_r(i, j)}_{\text{Range kernel}} \cdot f(x - i, y - j)$$

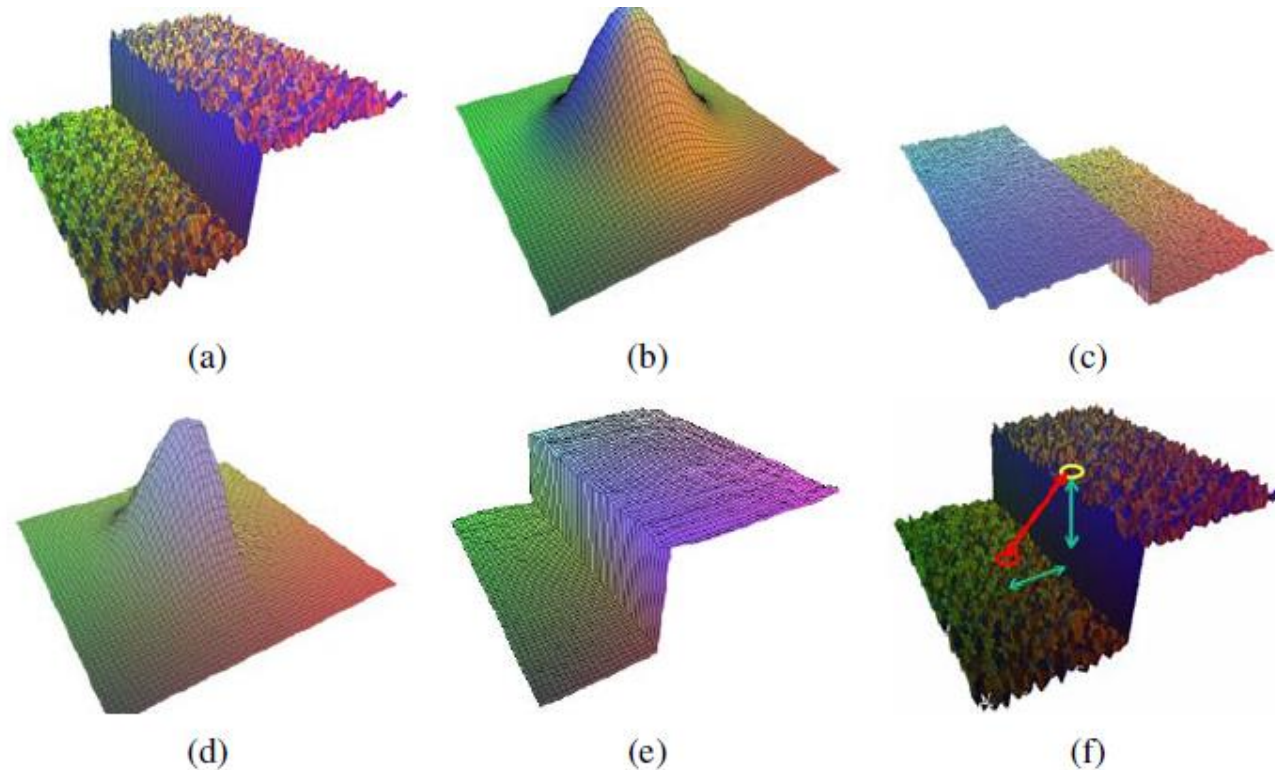
- **Spatial kernel:** weights are larger for pixels near the window center

$$h_s(i, j) = e^{-\frac{i^2 + j^2}{2\sigma_s^2}}$$

- **Range kernel:** weights are larger if the neighbor pixel has similar intensity (color) to the center pixel

$$h_r(i, j) = e^{-\frac{(f(x-i, y-j) - f(x, y))^2}{2\sigma_r^2}}$$

# Bilateral Filtering



**Figure 3.20** Bilateral filtering (Durand and Dorsey 2002) © 2002 ACM: (a) noisy step edge input; (b) domain filter (Gaussian); (c) range filter (similarity to center pixel value); (d) bilateral filter; (e) filtered step edge output; (f) 3D distance between pixels.



# Joint Bilateral Filtering

- The range kernel takes another **guidance image** as reference

$$g(x, y) = \frac{1}{W} \sum_{i, j \in [-r, r]} h_s(i, j) \cdot h_r(i, j) \cdot f(x - i, y - j)$$

$$h_s(i, j) = e^{-\frac{i^2 + j^2}{2\sigma_s^2}} \quad h_r(i, j) = e^{-\frac{(f'(x-i, y-j) - f'(x, y))^2}{2\sigma_r^2}}$$



Flash



No flash



Joint bilateral filtering



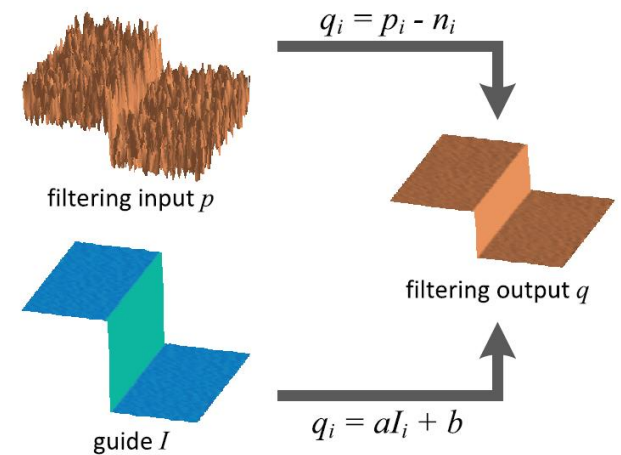
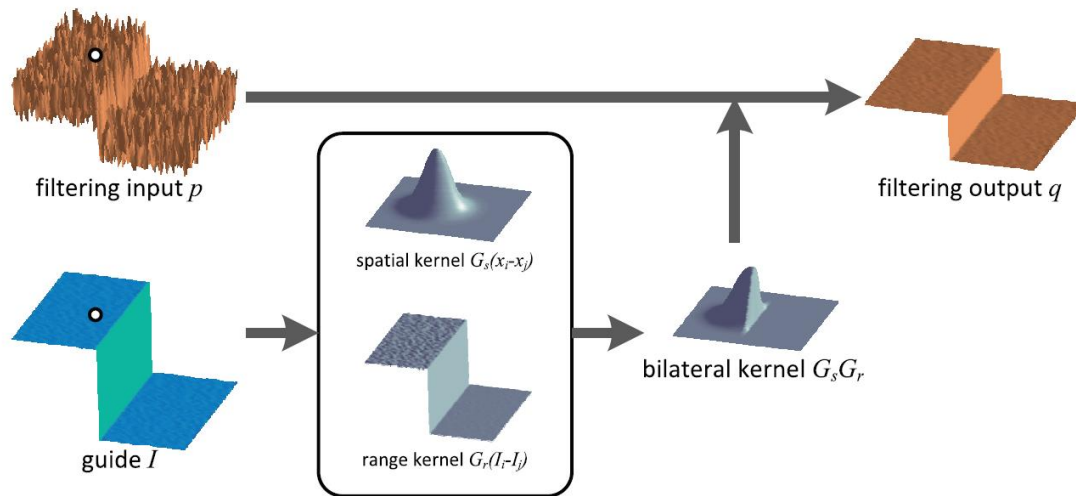
# Bilateral Filtering

- Fast bilateral filtering also exists
  - “A fast approximation of the bilateral filter using a signal processing approach”, ECCV 2006
  - “Constant time  $O(1)$  bilateral filtering”, CVPR 2008
  - “Real-time  $O(1)$  bilateral filtering”, CVPR 2009
  - “Fast high-dimensional filtering using the permutohedral lattice”, EG 2010
  - and many more...
- We also contribute some works in this field
  - “Constant time bilateral filtering for color images”, ICIP 2016
  - “VLSI architecture design of layer-based bilateral and median filtering for 4k2k videos at 30fps”, ISCAS 2017

# Guided Image Filtering

[He et al. ECCV 2010]

- Local linear assumption
- Per pixel  $O(1)$



# Guided Image Filtering

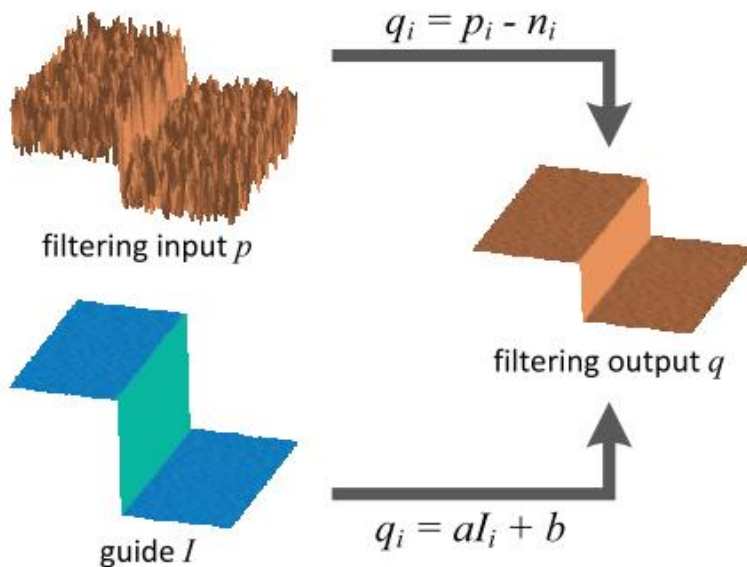
[He et al. ECCV 2010]

- Local linear assumption

$$E(a_k, b_k) = \sum_{i \in \omega_k} ((a_k I_i + b_k - p_i)^2 + \epsilon a_k^2). \xrightarrow{\text{Linear regression}}$$

$$a_k = \frac{\frac{1}{|\omega|} \sum_{i \in \omega_k} I_i p_i - \mu_k \bar{p}_k}{\sigma_k^2 + \epsilon},$$

$$b_k = \bar{p}_k - a_k \mu_k.$$



## Algorithm 1. Guided Filter.

**Input:** filtering input image  $p$ , guidance image  $I$ , radius  $r$ , regularization  $\epsilon$

**Output:** filtering output  $q$ .

- 1:  $\text{mean}_I = f_{\text{mean}}(I)$   
 $\text{mean}_p = f_{\text{mean}}(p)$   
 $\text{corr}_I = f_{\text{mean}}(I * I)$   
 $\text{corr}_{Ip} = f_{\text{mean}}(I * p)$
- 2:  $\text{var}_I = \text{corr}_I - \text{mean}_I * \text{mean}_I$   
 $\text{cov}_{Ip} = \text{corr}_{Ip} - \text{mean}_I * \text{mean}_p$
- 3:  $a = \text{cov}_{Ip} / (\text{var}_I + \epsilon)$   
 $b = \text{mean}_p - a * \text{mean}_I$
- 4:  $\text{mean}_a = f_{\text{mean}}(a)$   
 $\text{mean}_b = f_{\text{mean}}(b)$
- 5:  $q = \text{mean}_a * I + \text{mean}_b$

# Edge-Preserving Filtering

- Both the bilateral filter and the guided image filter are called **edge-preserving filters** (EPF)
  - They can smooth images while preserving edges
- Other widely used EPFs with source code
  - [Weighted least square filter](#), SIGGRAPH 2008
  - [Domain transform filter](#), SIGGRAPH 2011
  - [\$L\_0\$  filter](#), SIGGRAPH Asia 2011
  - [Fast global smoothing filter](#), TIP 2014

# Edge-Preserving Filtering

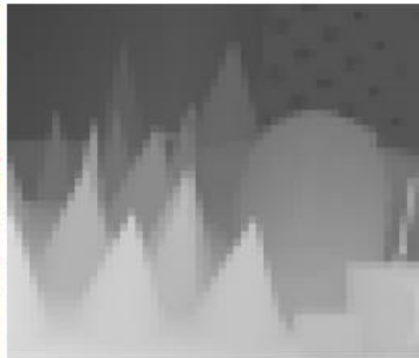
- Applications



Edit (e.g. color) propagation



Detail manipulation



Guided upsampling (depth maps, features, ..., etc.)

# What we have learned today

- Digital imaging
- Some low-level image processing

Physical world

Sensing device

Interpreting device

Interpretations



CPU/GPU/DSP

cat, lovely,  
in a box

Image formation

- Pinhole camera
- Photography
- Camera pipeline

Image processing

- Histogram
- Morphological operations
- Edge detection
- Image filtering
- ...