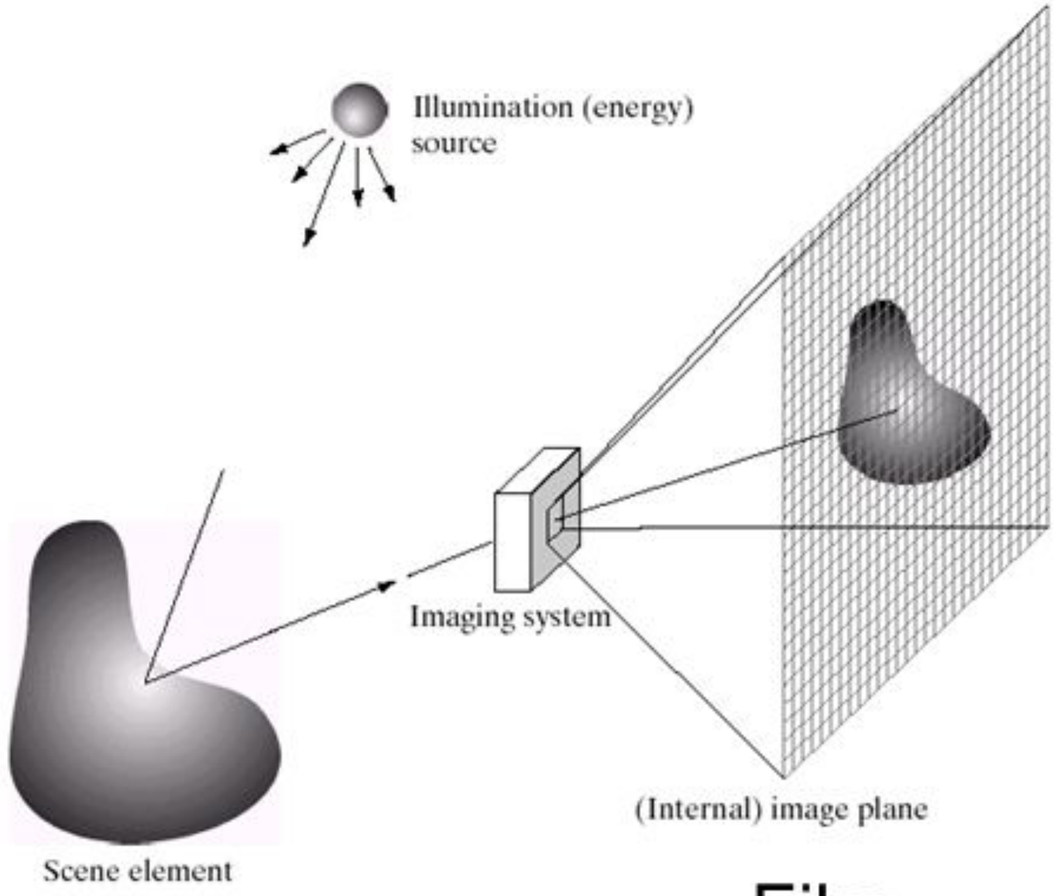
TDT4265:
Computer Vision & Deep Learning

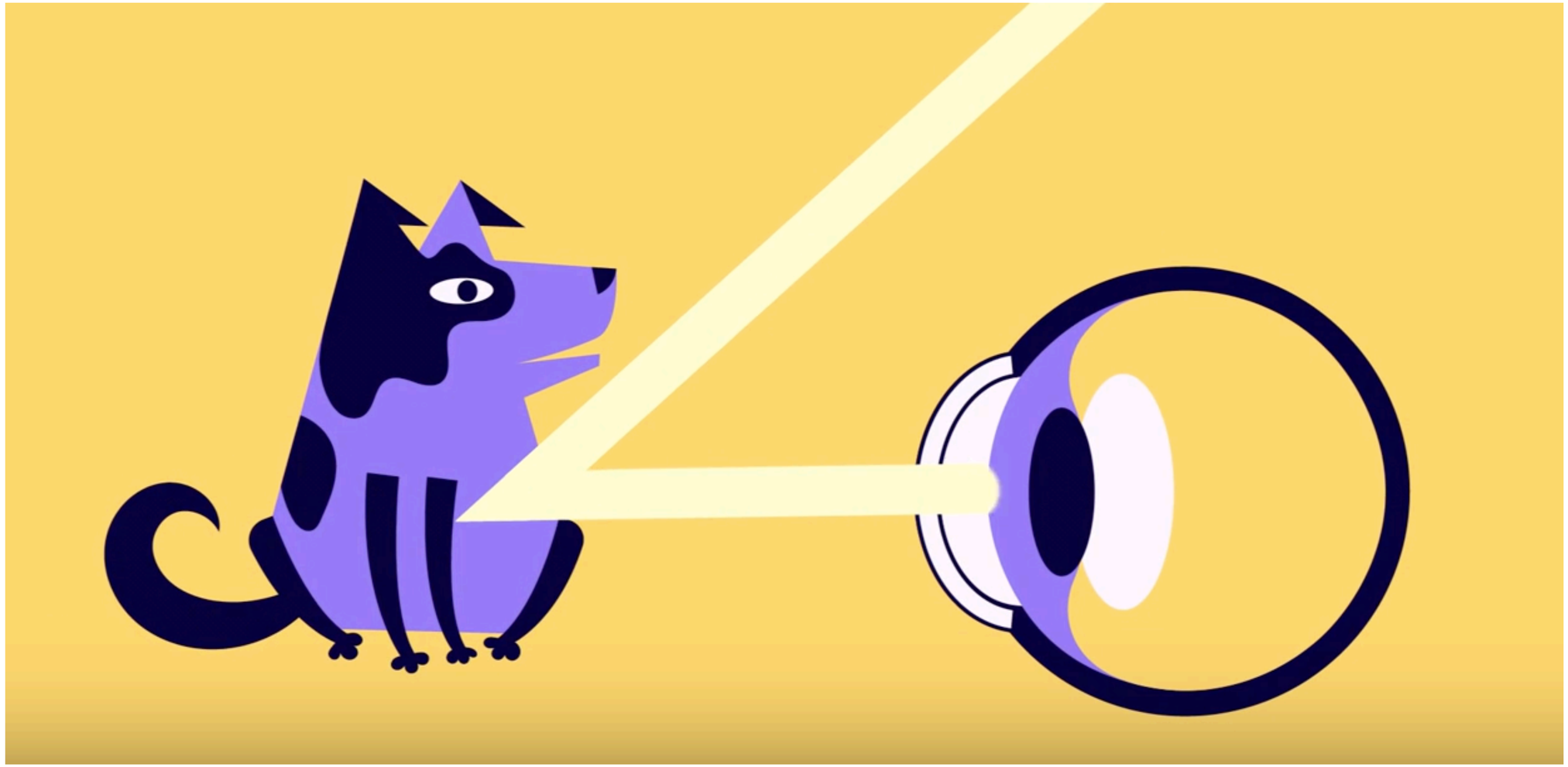
Intro

Frank Lindseth, IDI, IE, NTNU

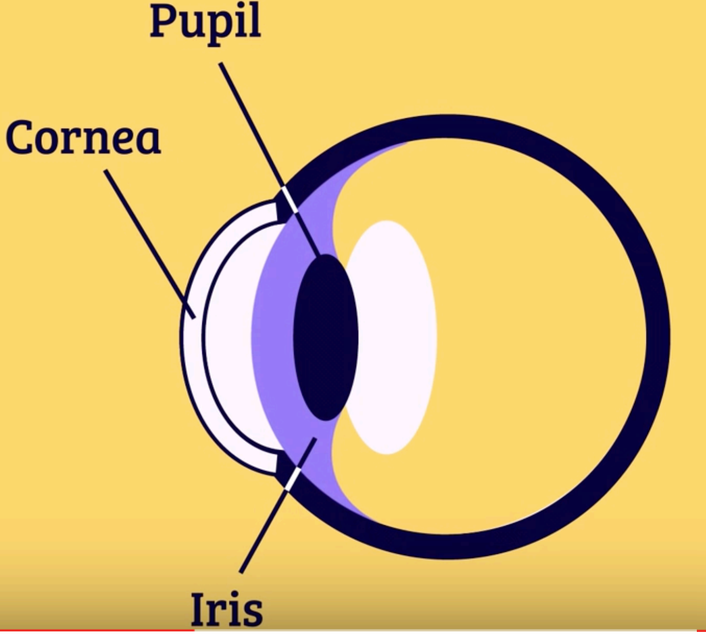
Image formation & Imaging systems

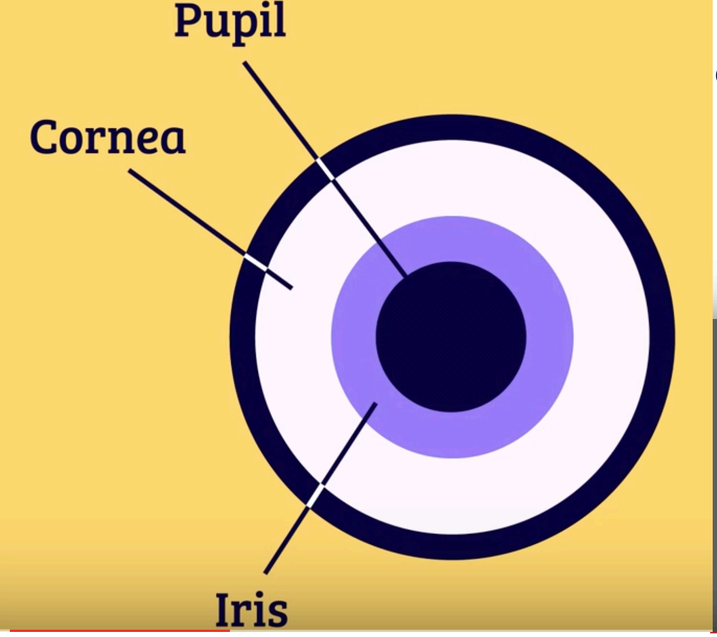


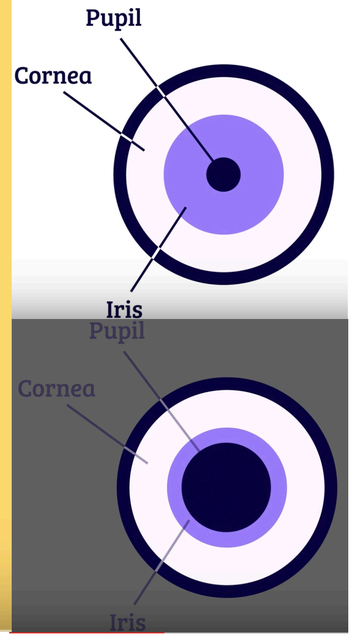
Human Eyes

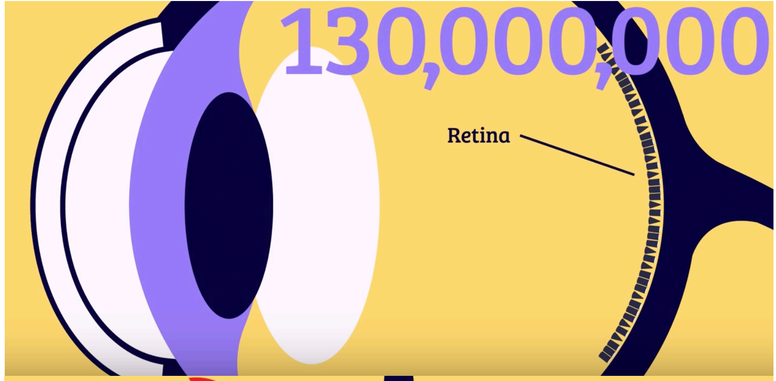


The Visual System: How Your Eyes Work


Pupil
Cornea
Iris


Pupil
Cornea
Iris


Pupil
Cornea
Iris
Pupil
Cornea
Iris


130,000,000
Retina

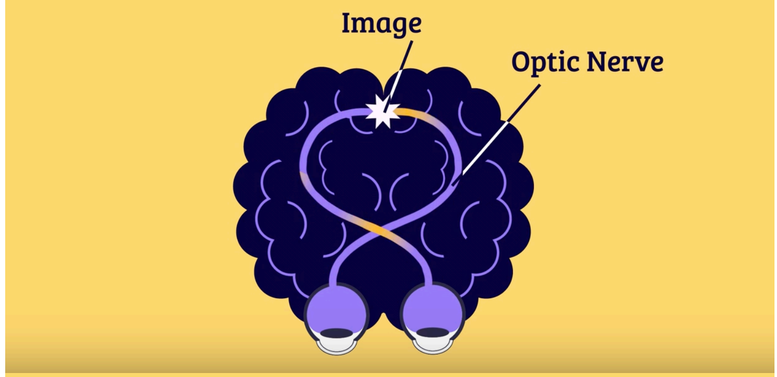
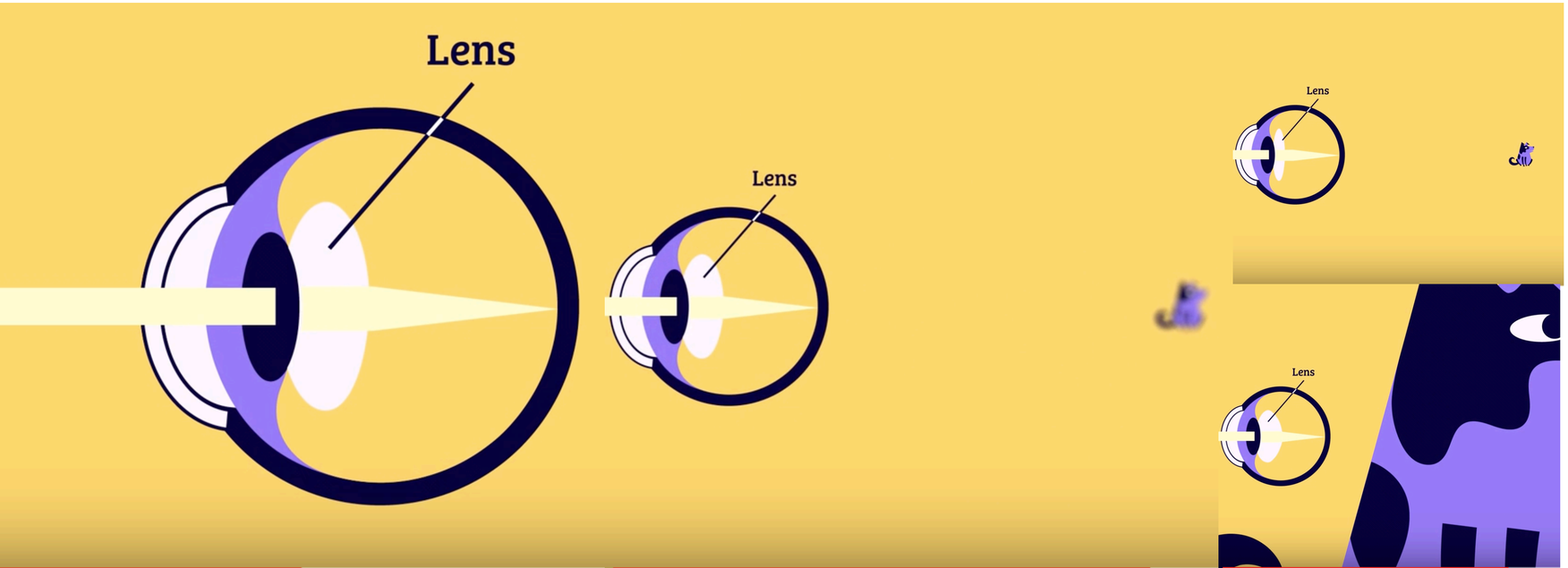
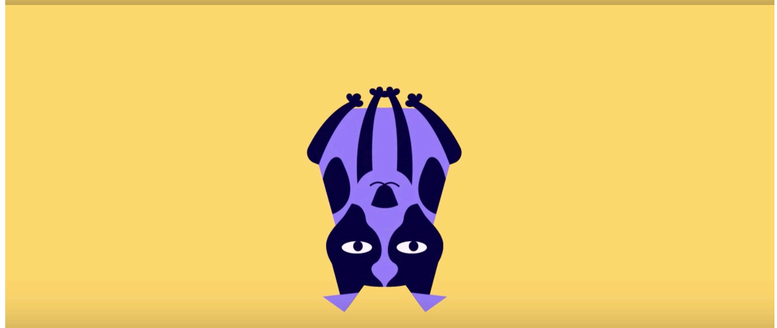


Image
Optic Nerve


Lens
Lens

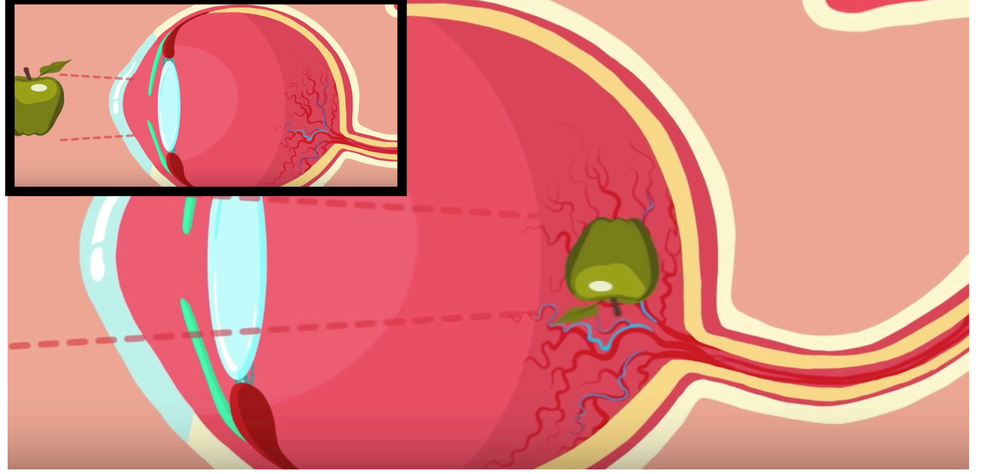
Lens
Lens



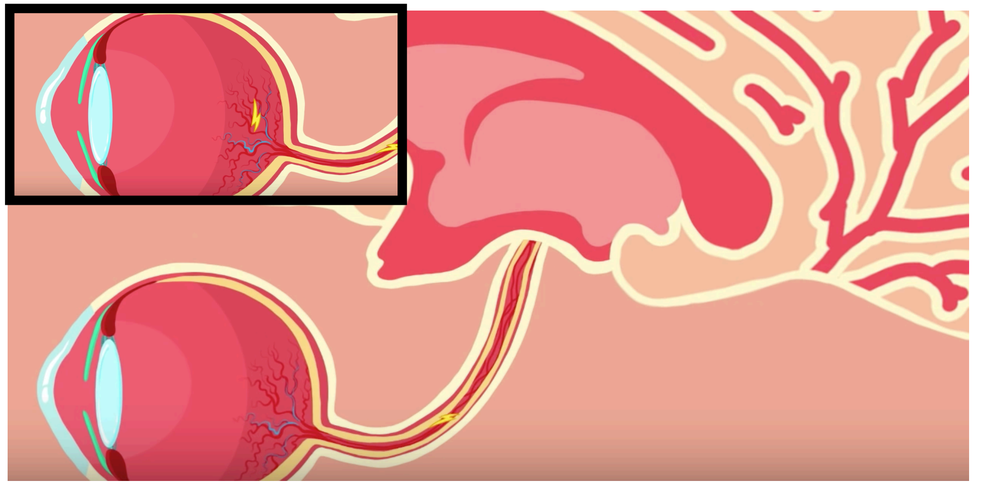
Human Eyes (2)

How Your Eyes Work (2)

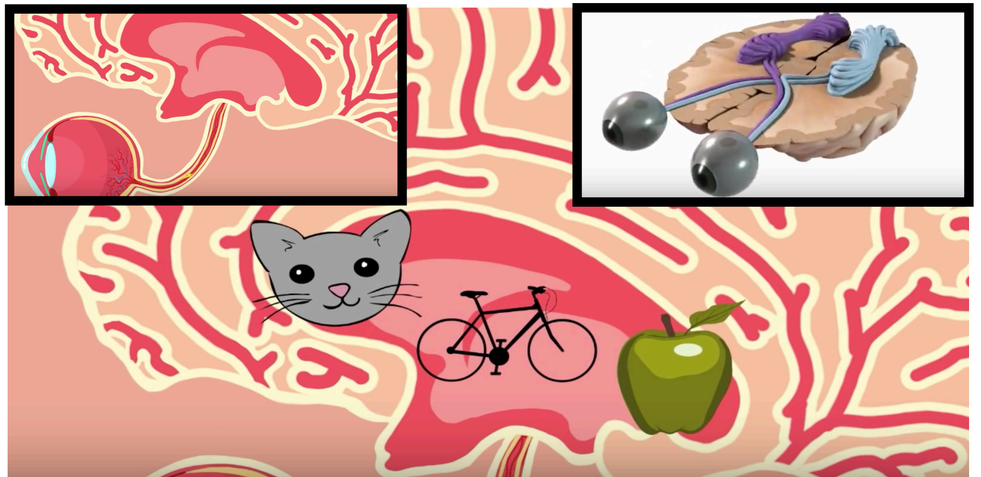

RETINA
120 MILLION RODS
7 MILLION CONES




ROD VISION




CONE VISION



Human Eyes (3)

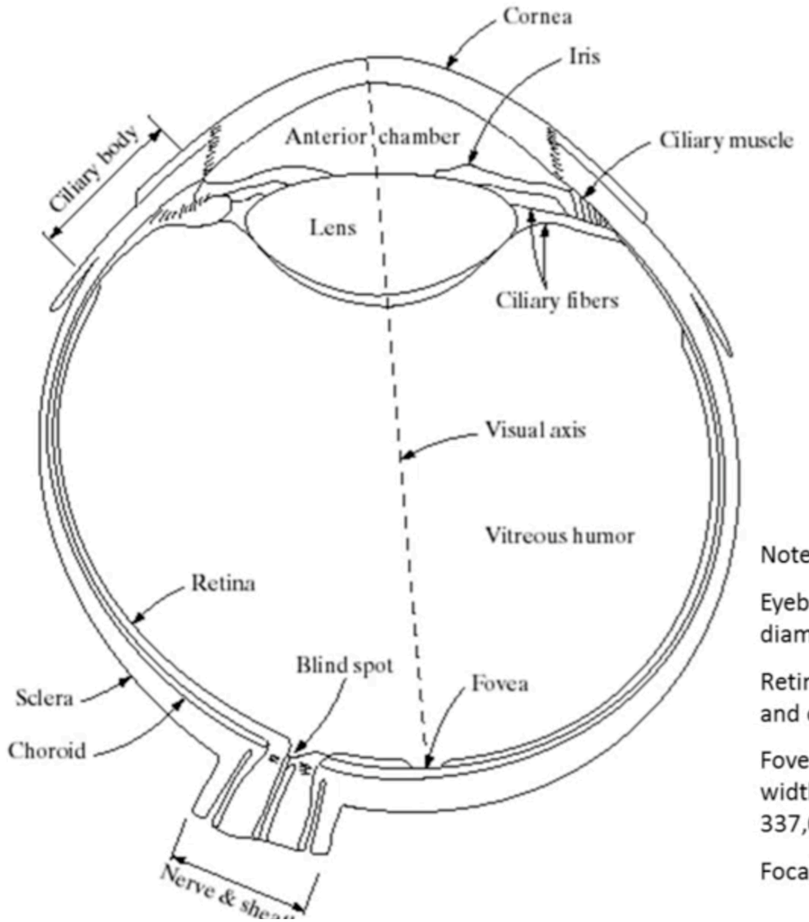


FIGURE 2.1
Simplified diagram of a cross section of the human eye.

Notes:

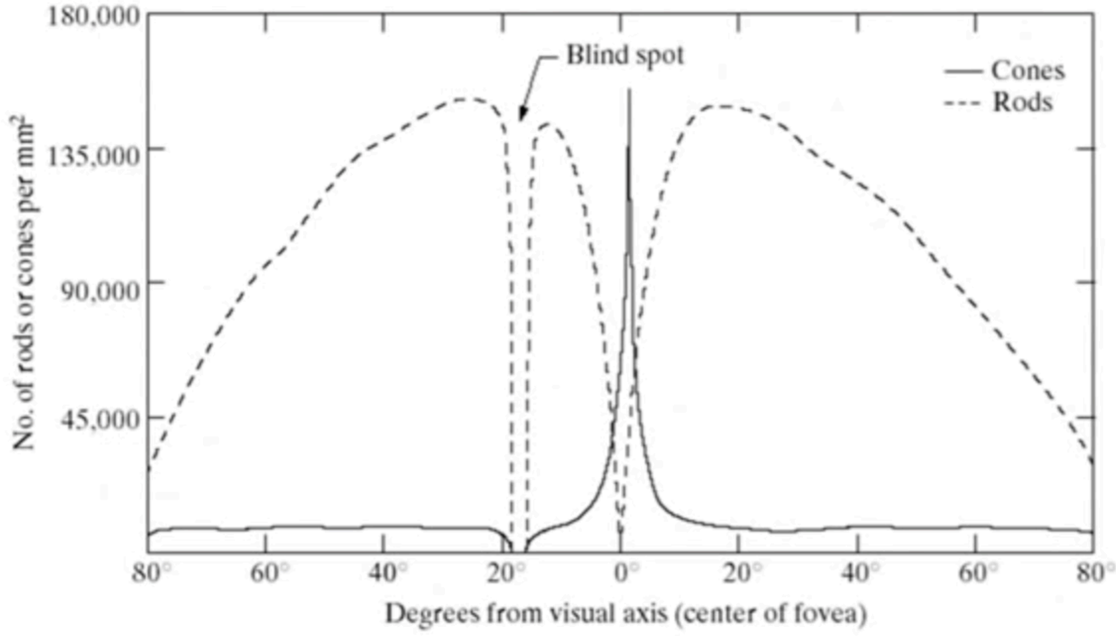
Eyeball is about 20 mm in diameter

Retina contains both rods and cones

Fovea is about 1.5 mm in width, contains about 337,000 cones

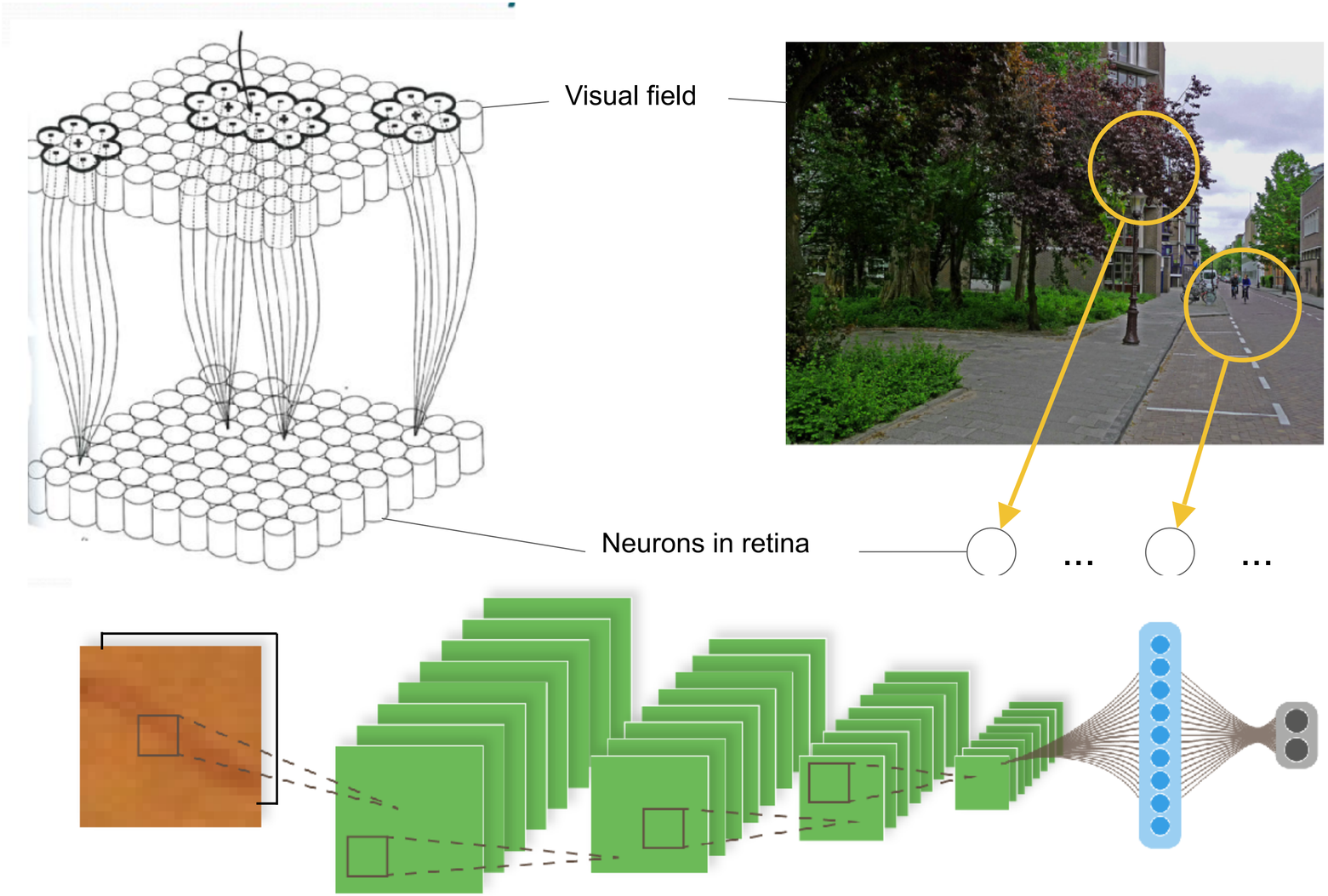
Focal length about 17 mm

- Rods sensitive to low light (scotopic vision)
- Cones detect color, work in bright light (photopic vision)

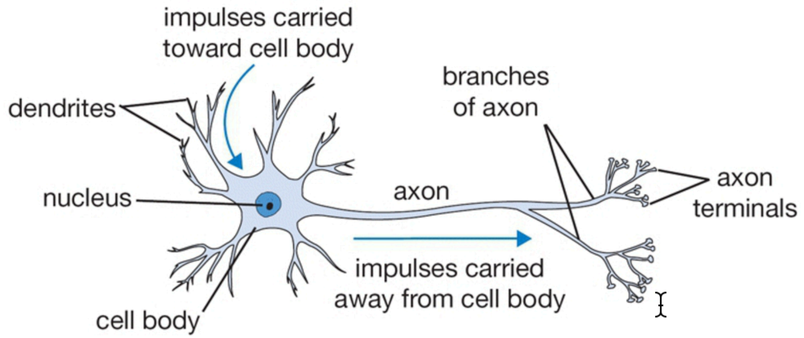


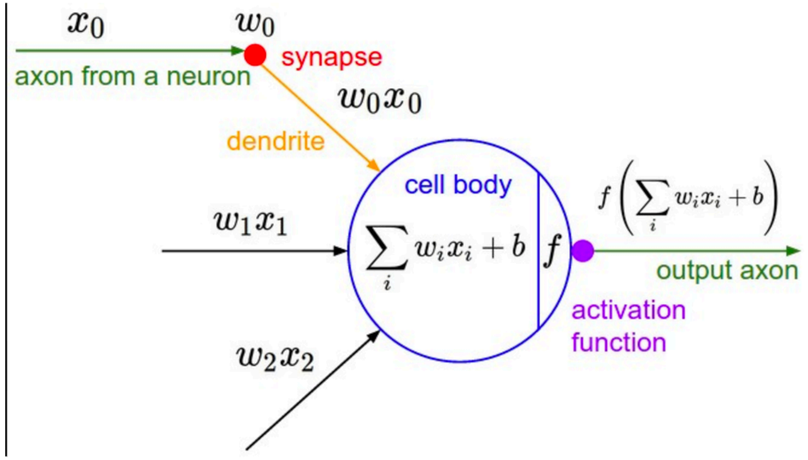
Biological receptive fields

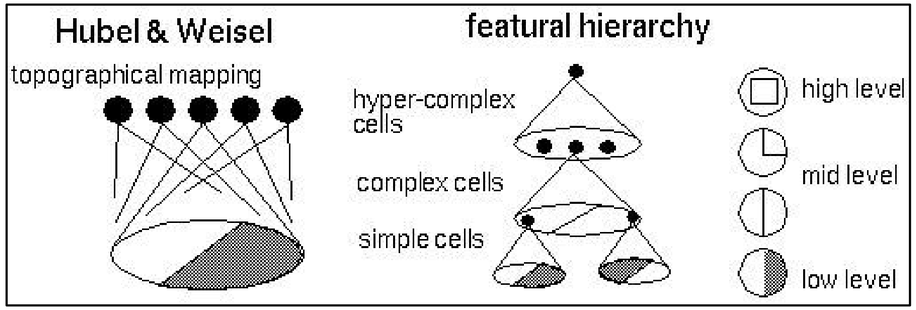
- Specific neurons in the retina cares about only a small part of the visual field



Biological vs. Artificial Neuron and Network







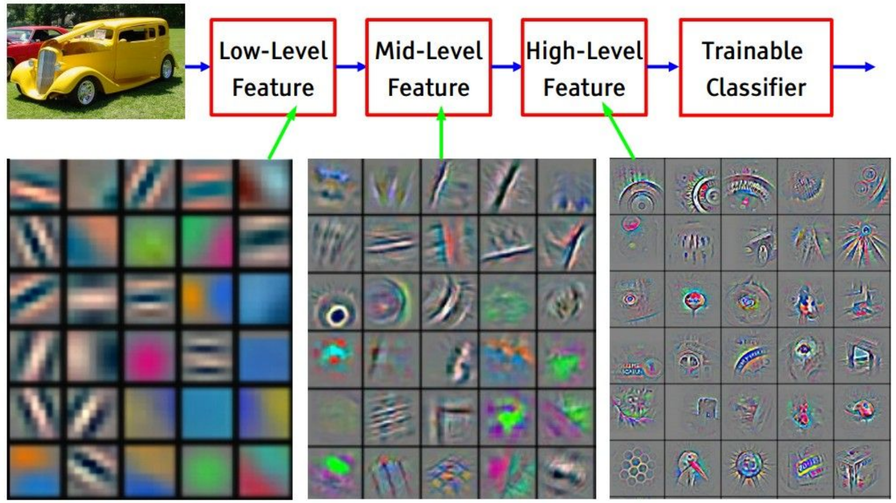
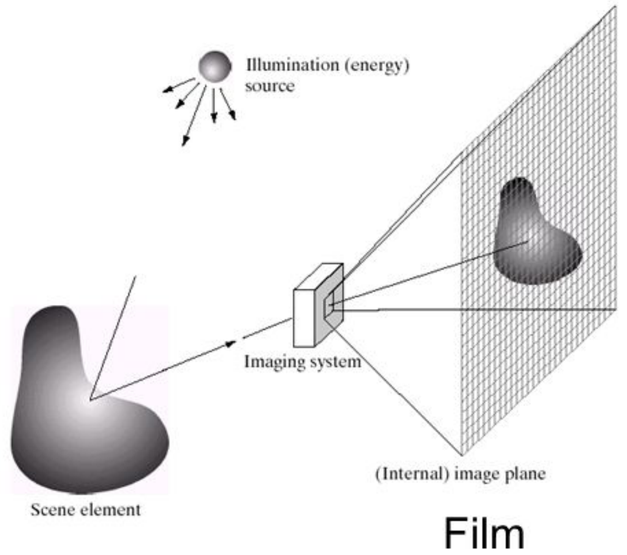
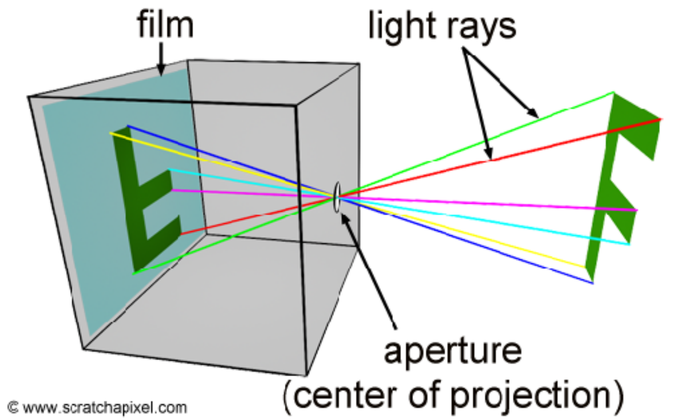


Image formation & the Pinhole camera

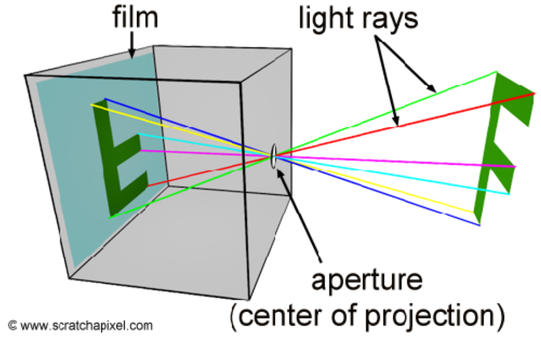
- The **image** I(x,y) measures how much light that is captured at pixel (x,y)

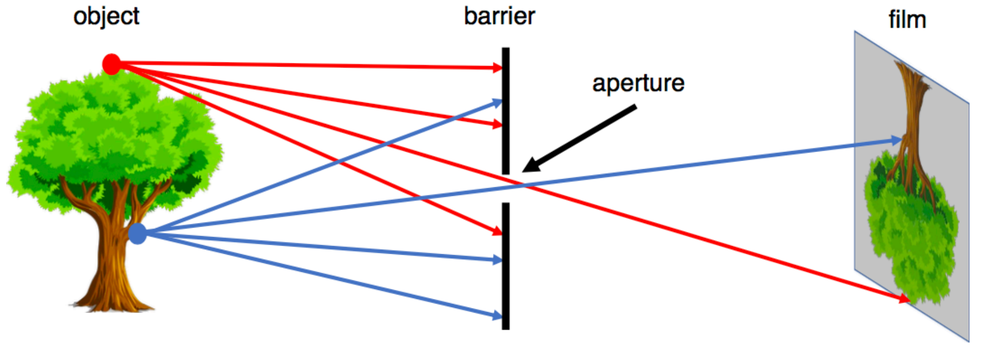


- We want to know:
  - Where does a point P(X,Y, Z) in the world get imaged?
  - What is the brightness at the resulting point p(x,y)?



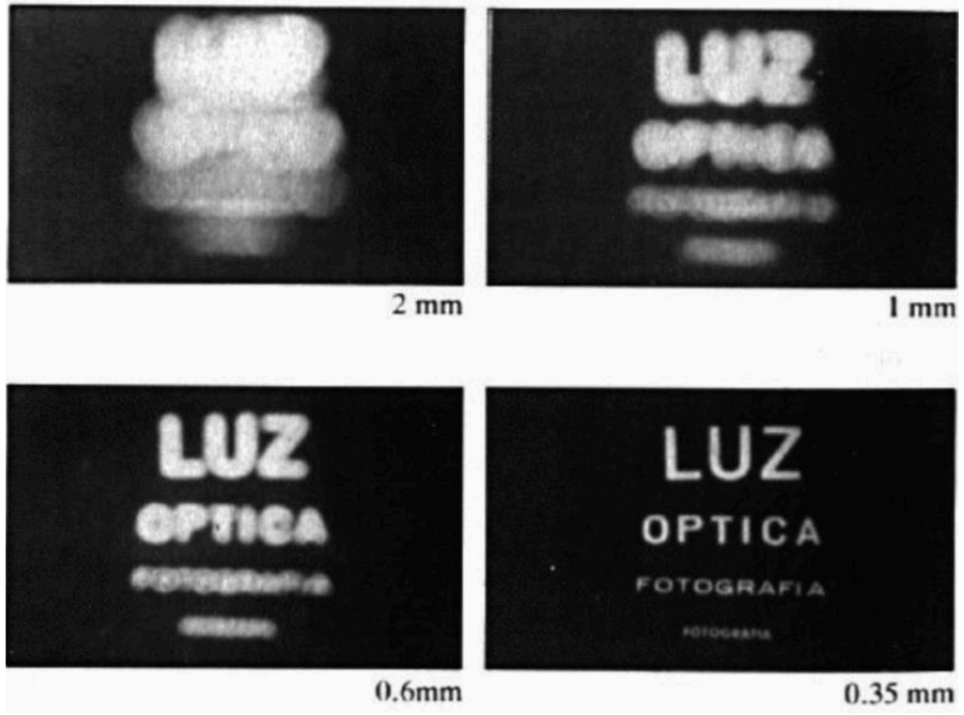
The Pinhole camera model (2)



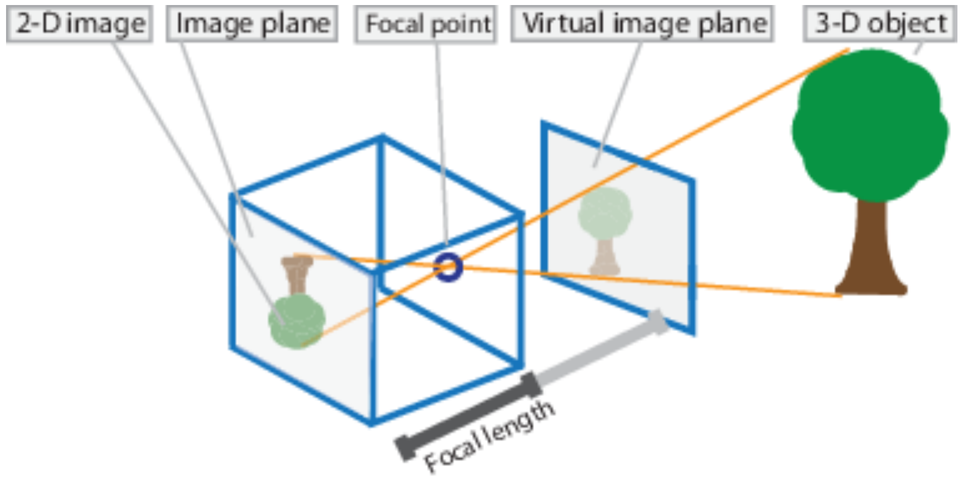


- **Without** a barrier in place, *every* point on the film will be influenced by light rays reflected from *every* point on the 3D object. Due to the **barrier**, only *one* (or a few) of these rays of light passes through the aperture, hits the film and generates an inverted image

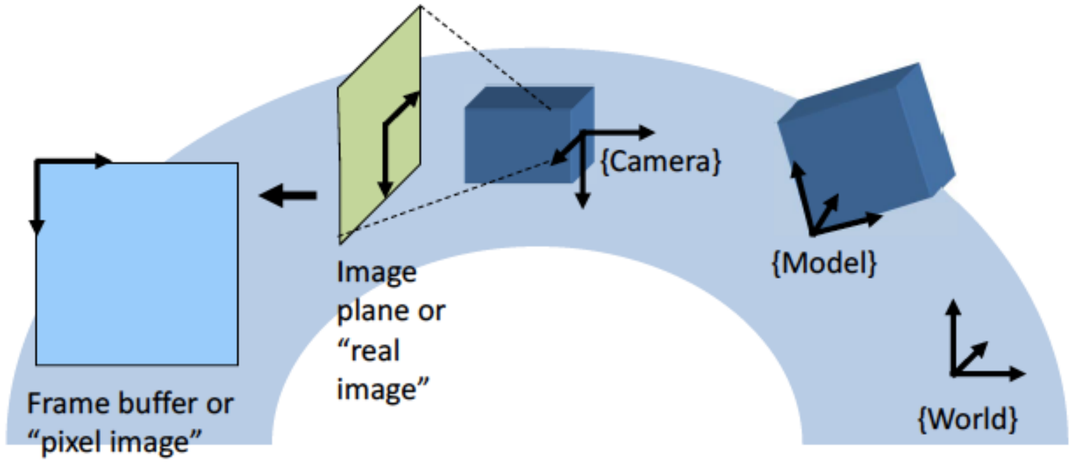
- As the **aperture** size **decreases**, the image gets **sharper, but darker**.

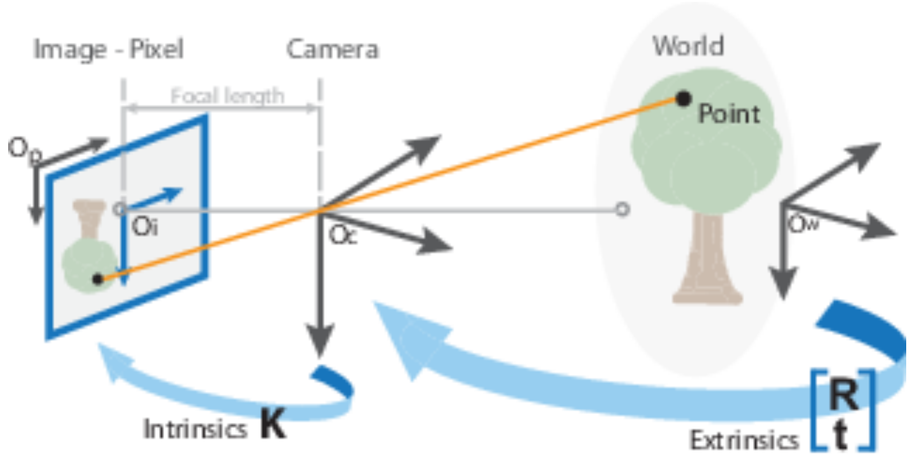


The Pinhole camera model (3)



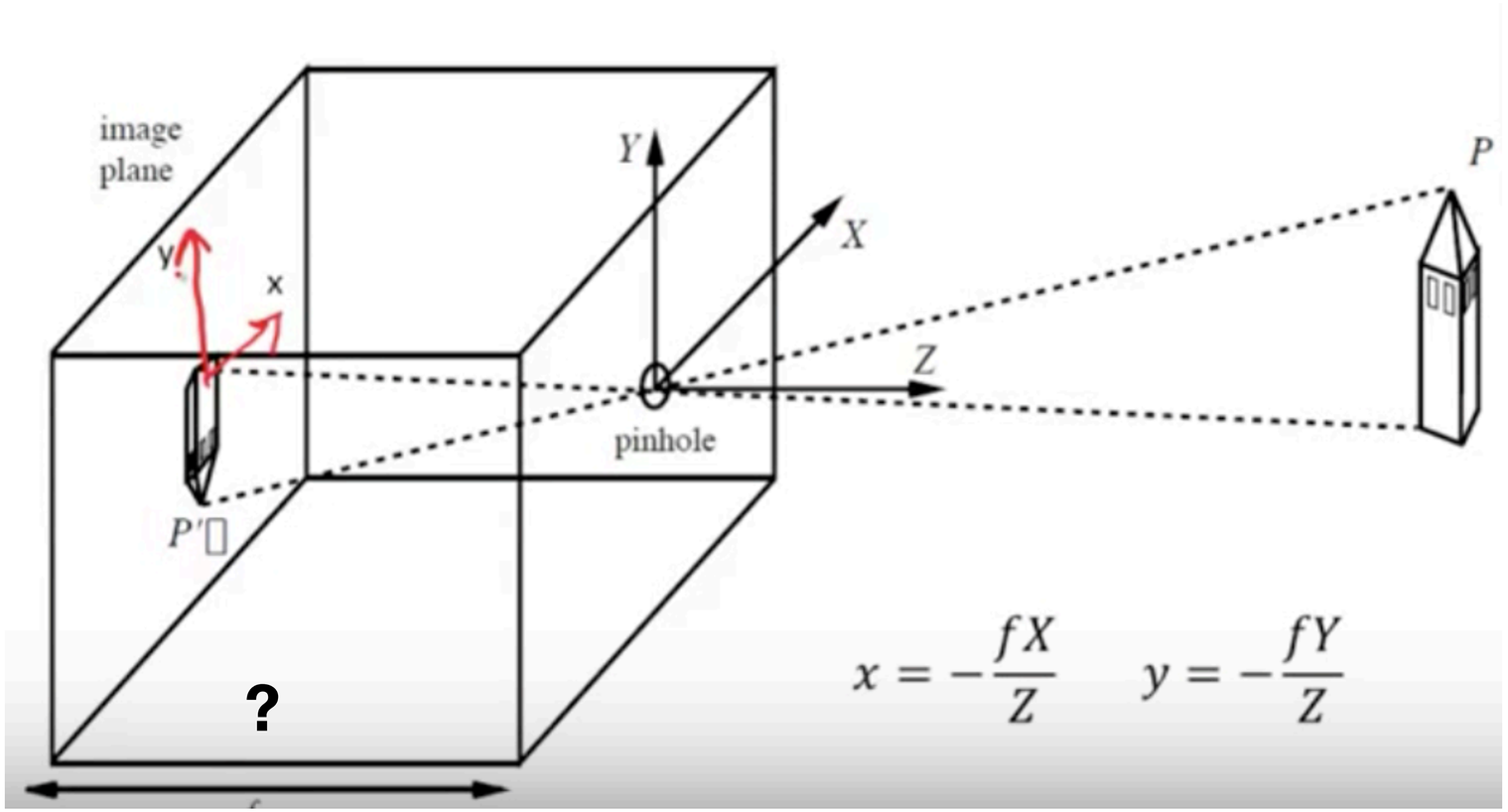
Simple camera without a lens and with a single small aperture. Light rays pass through the aperture and project an inverted image.





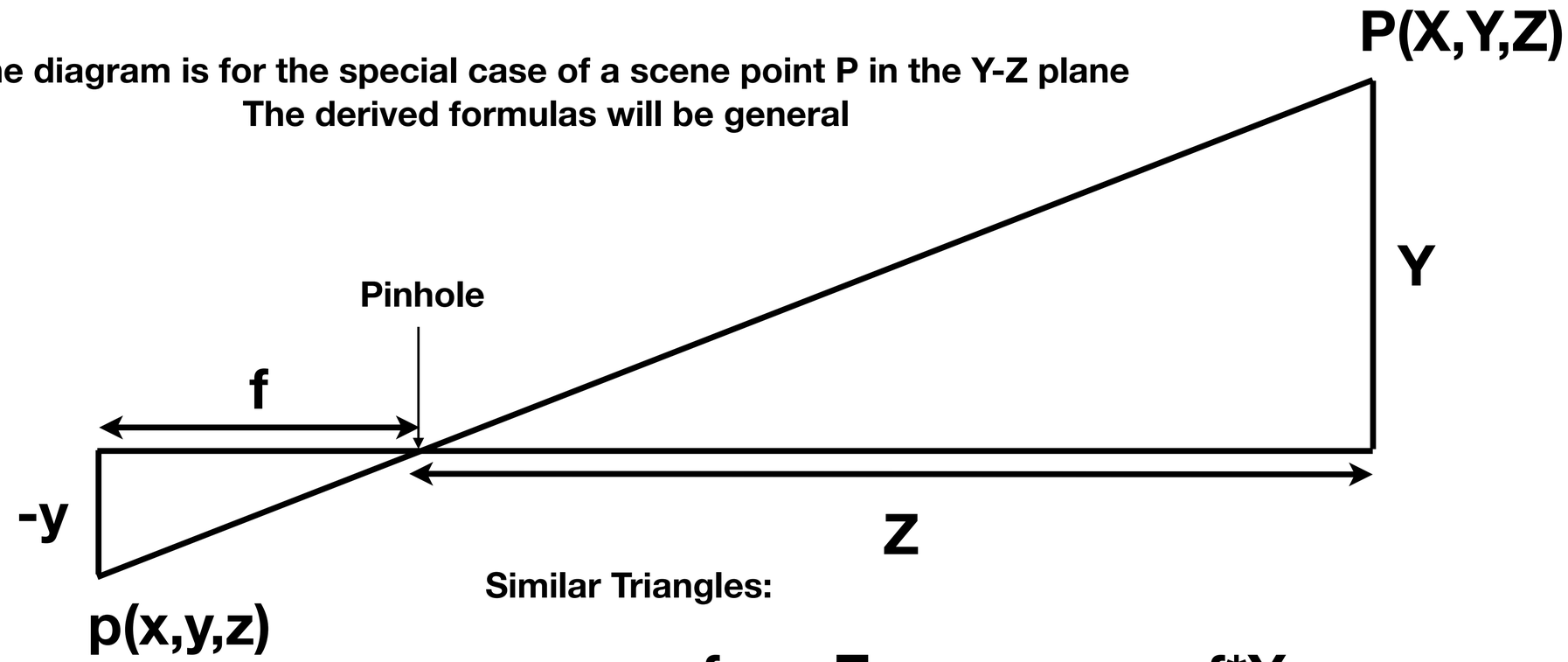
Extrinsic parameters:
world coordinates -> camera coordinates
Intrinsics parameters:
camera coordinates -> image plane coordinates

The Pinhole camera model (4)



The Pinhole camera model (5)

The diagram is for the special case of a scene point P in the Y-Z plane
The derived formulas will be general



Similar Triangles:

$$\frac{f}{-y} = \frac{Z}{Y} \quad \Rightarrow \quad y = -\frac{f*Y}{Z}$$

Example

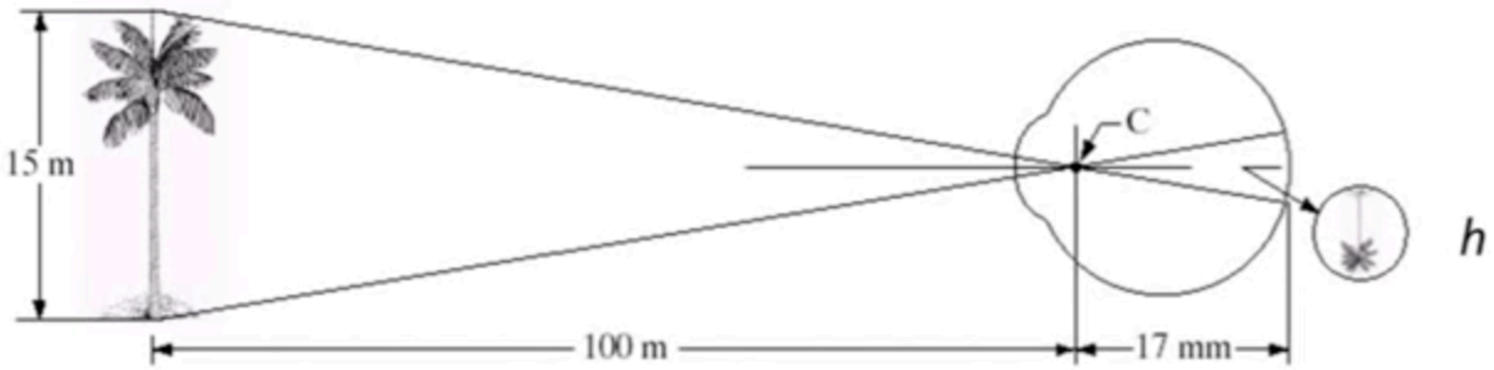
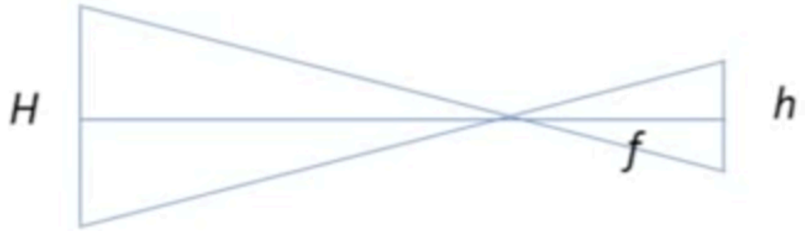


FIGURE 2.3
Graphical representation of the eye looking at a palm tree. Point *C* is the optical center of the lens.

- Treat as pinhole camera – good approximation
  - Similar triangles



- What is the size of the tree on the retina?

Example (2)

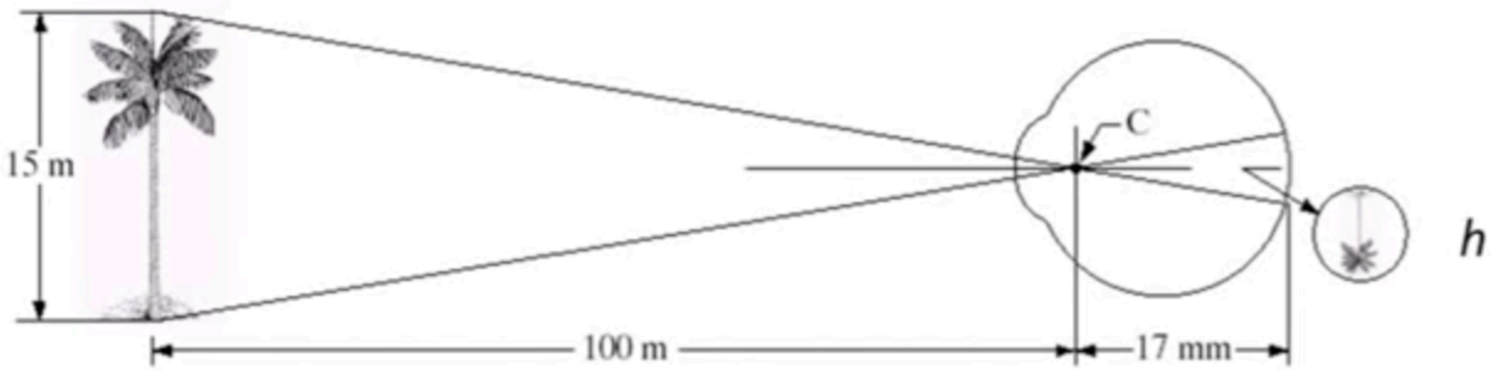
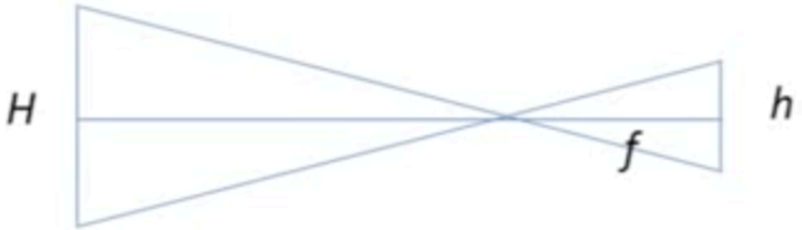


FIGURE 2.3
Graphical representation of the eye looking at a palm tree. Point *C* is the optical center of the lens.

- Treat as pinhole camera – good approximation
  - Similar triangles
- What is the size of the tree on the retina?
- What if the tree were twice as far away?

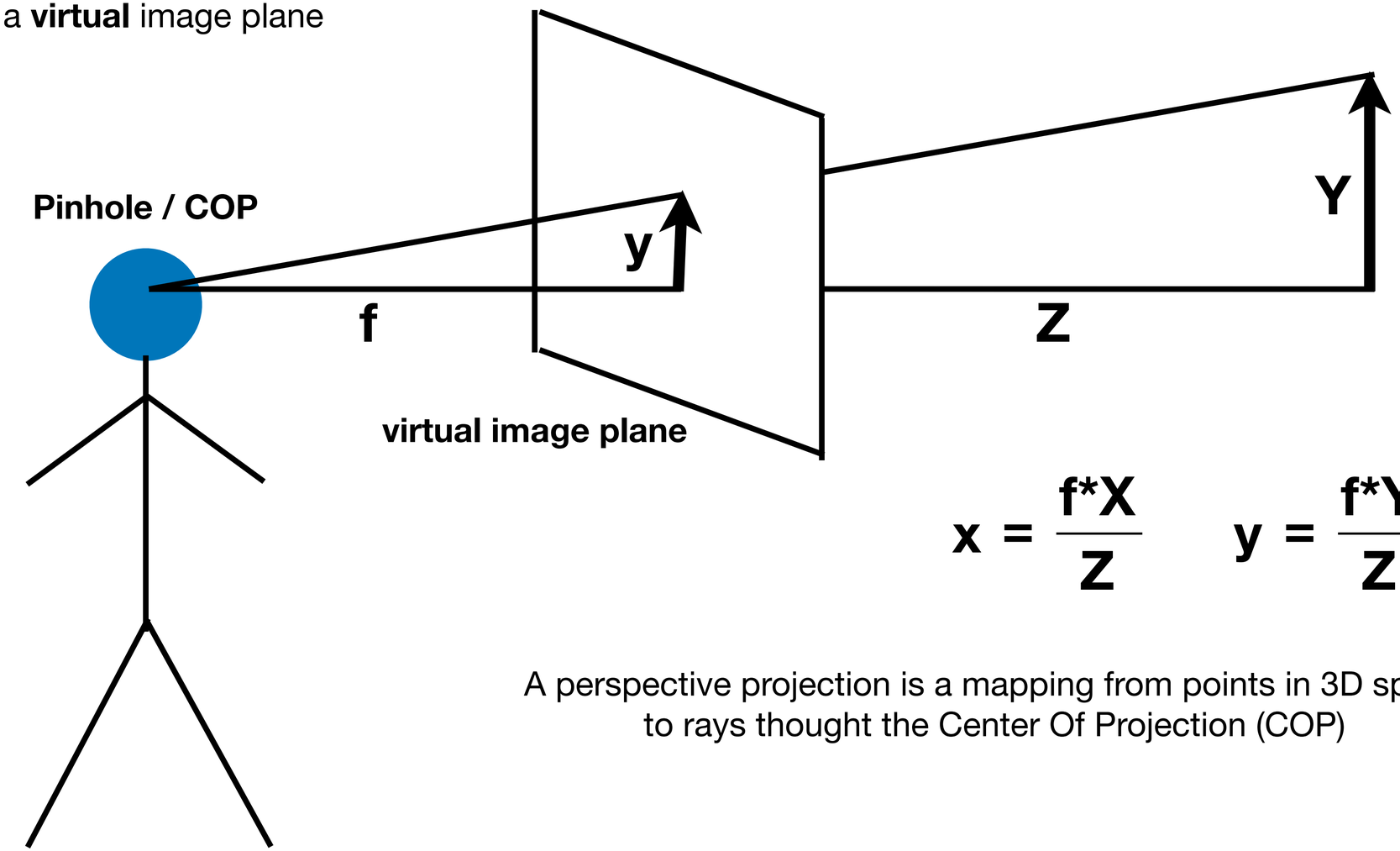


15/100 = h/17 or
h=2.55 mm

Y / Z = y / f

Pinhole camera: trick (no inversion)

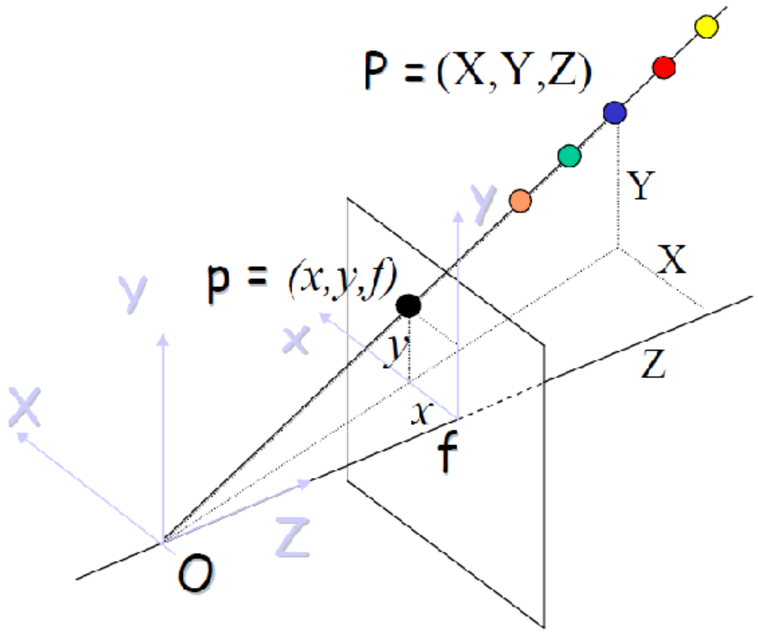
A projection model that avoids **inversion**
Trick: use a **virtual** image plane

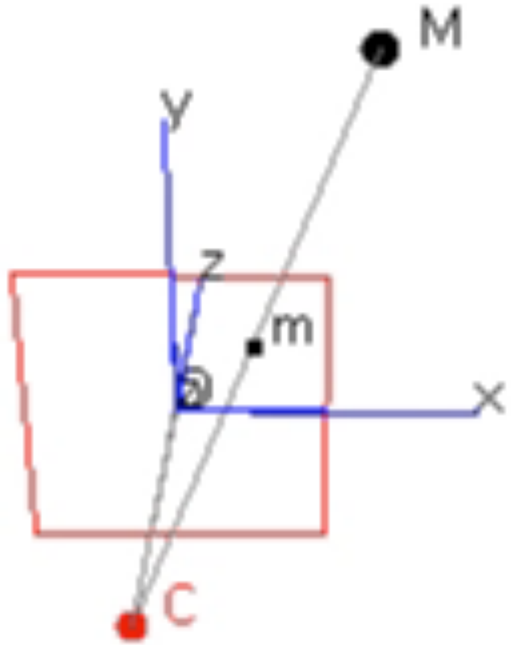


$$x = \frac{f*X}{Z} \qquad y = \frac{f*Y}{Z}$$

A perspective projection is a mapping from points in 3D space
to rays thought the Center Of Projection (COP)

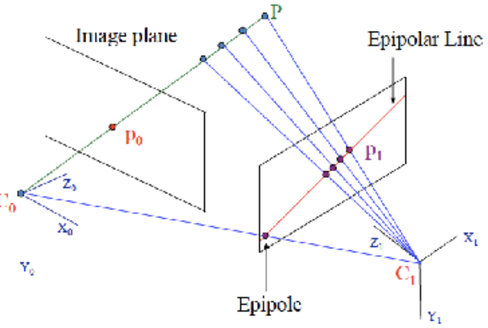
Single Camera / View

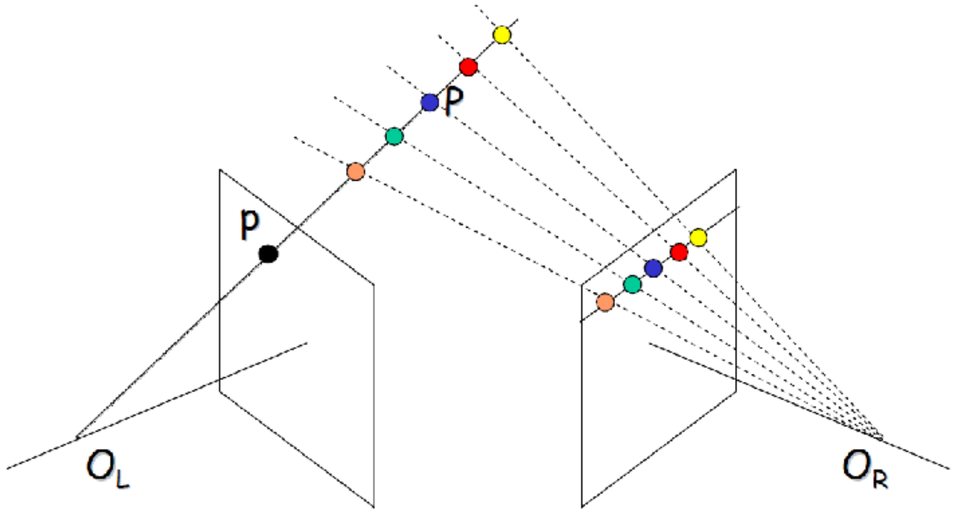


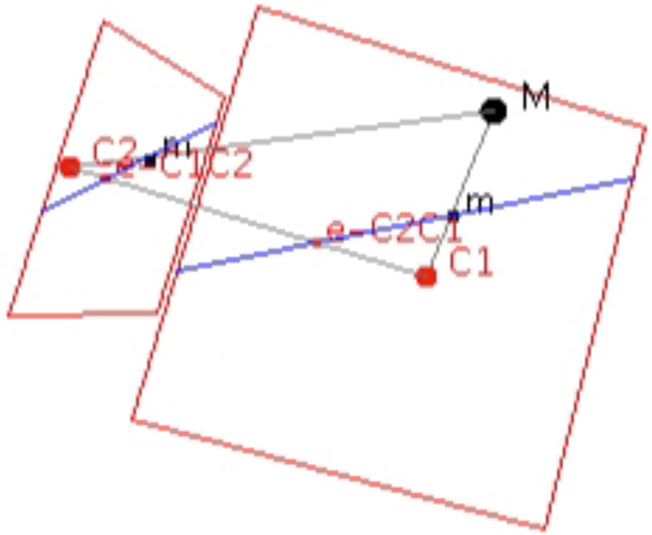


Fundamental ambiguity: Any scene point on the ray OP (CM) has image point p (m)

Stereo (multi-view)

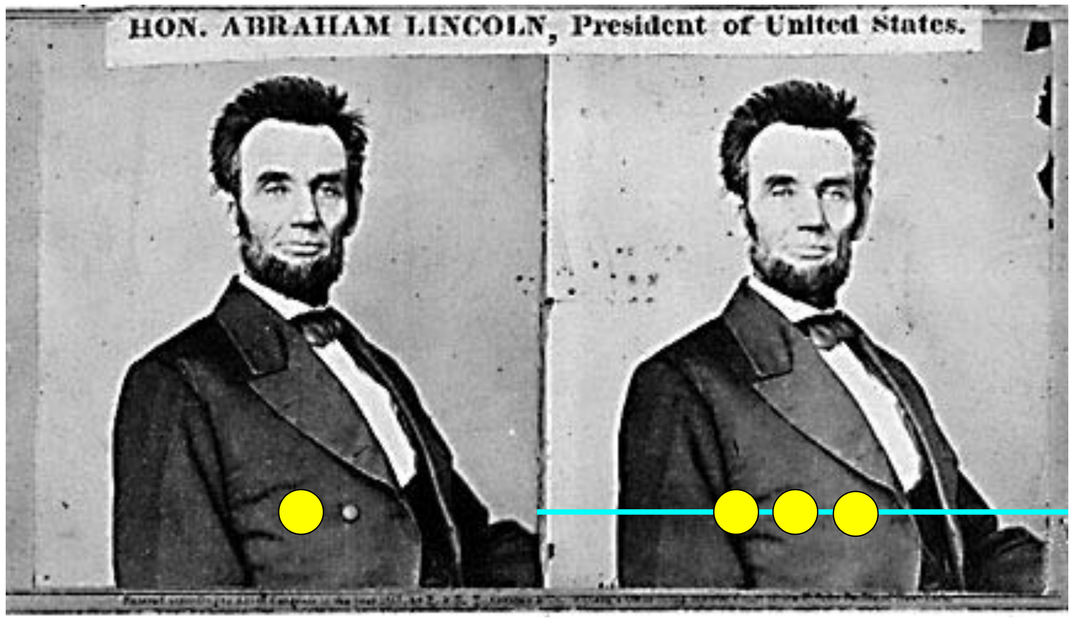






A second camera / view can **resolve the ambiguity**,
enabling measurements of **depth** via triangulation

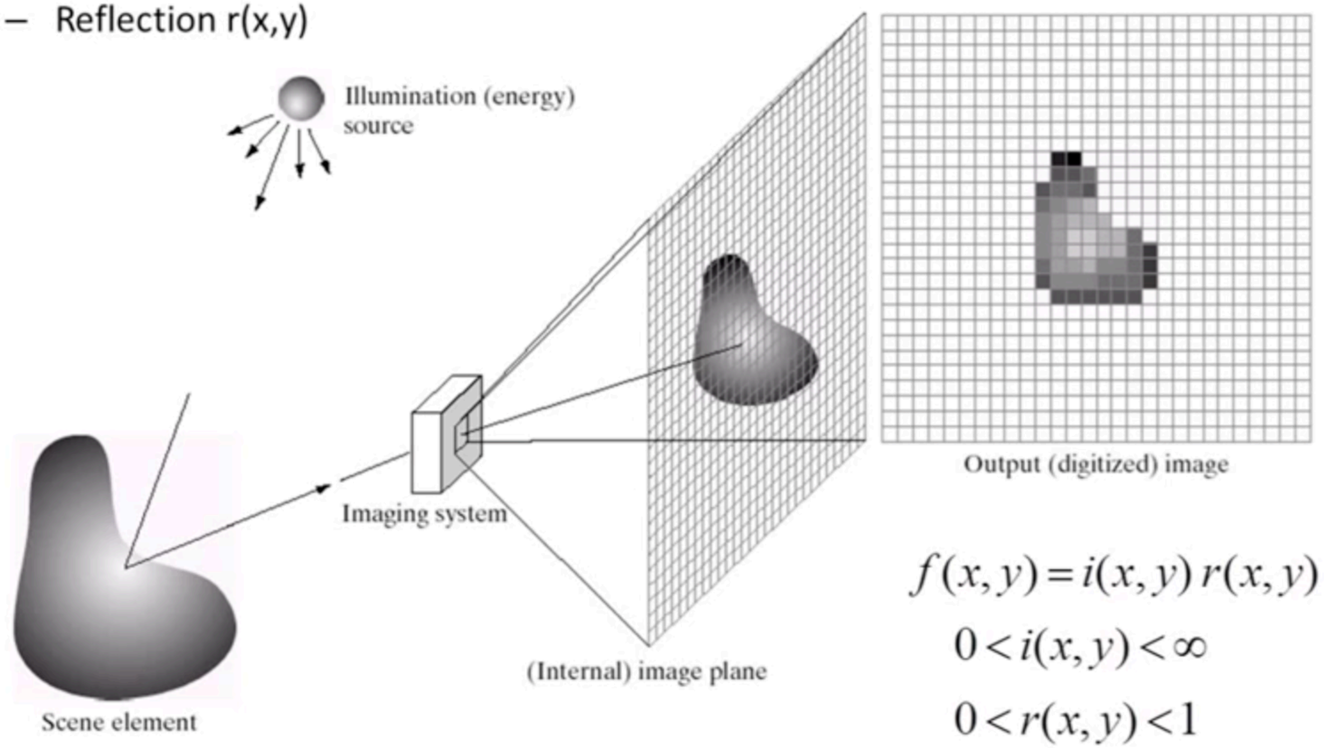
Stereo, correspondence, disparity and depth



- **For each pixel** x in the first image
  - Find corresponding epipolar **scanline** in the right image
  - Examine all pixels on the scanline and pick the best **match** x’
  - Compute **disparity** x-x’ and set **depth**(x) = b*f/(x-x’)

Digital Images

- Components
  - Illumination i(x,y)
  - Reflection r(x,y)



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

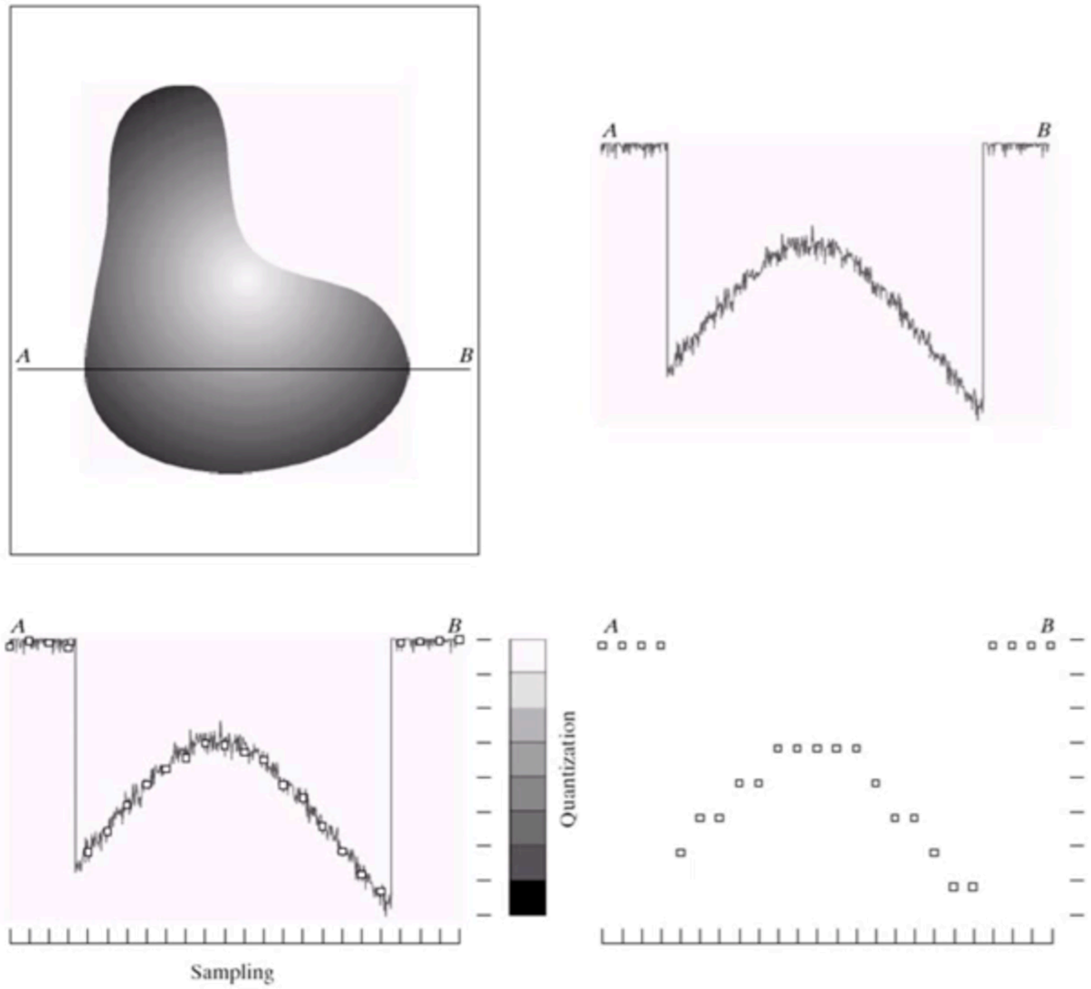
$$0 < i(x,y) < \infty$$

$$0 < r(x,y) < 1$$

a b c d e

FIGURE 2.15 An example of the digital image acquisition process. (a) Energy ("illumination") source. (b) An element of a scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

Sampling & Quantization

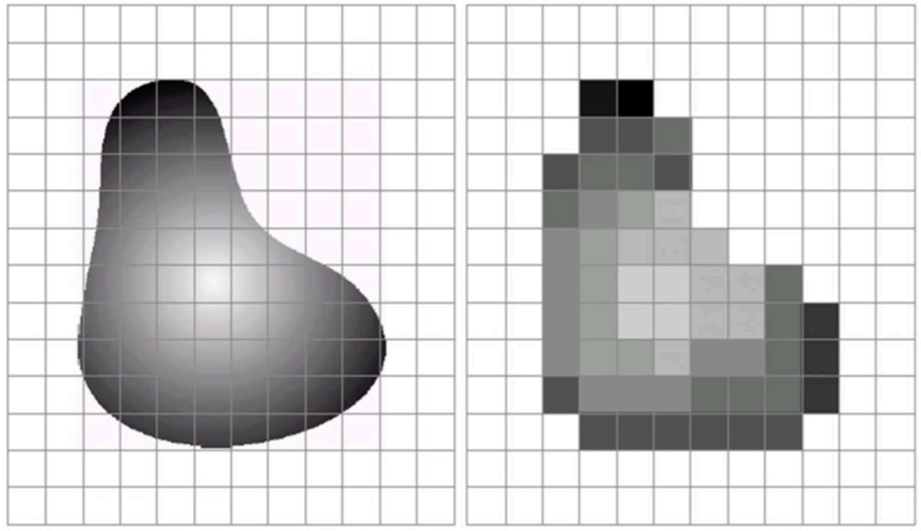


a b
c d

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

Sampling and Quantization

- N bits per pixel allows 2^N values



a b

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

Images as functions

An image can be thought of as:

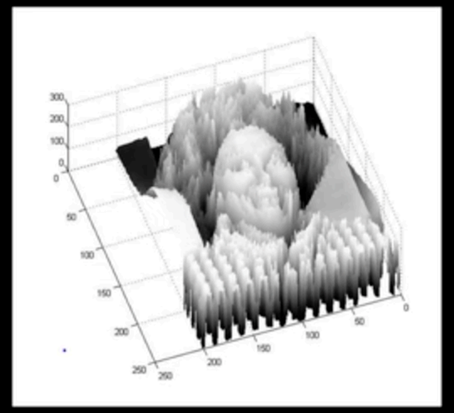
- A 2-dimensional array of numbers ranging from some minimum to some maximum
- A function I of x and y: $I(x, y)$
- Something generated by a camera.

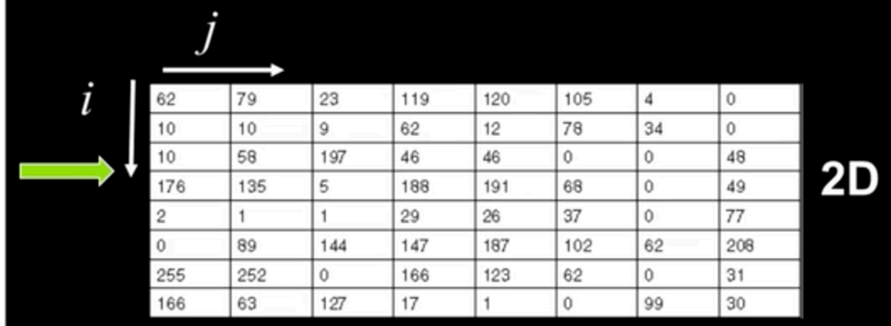
We think of an image as a *function*, f or I, from R^2 to R:

$f(x, y)$ gives the intensity or value at position (x, y)

Practically define the image over a rectangle, with a finite range:

$$f\colon [a, b]\ x\ [c, d] \rightarrow [min, max]$$





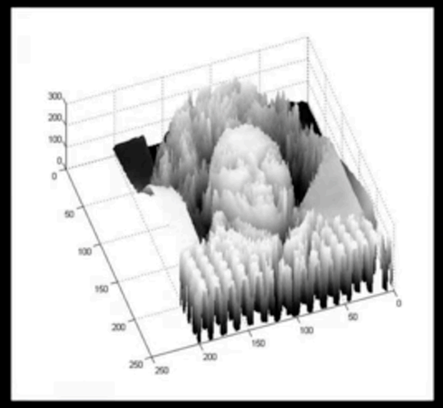
62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

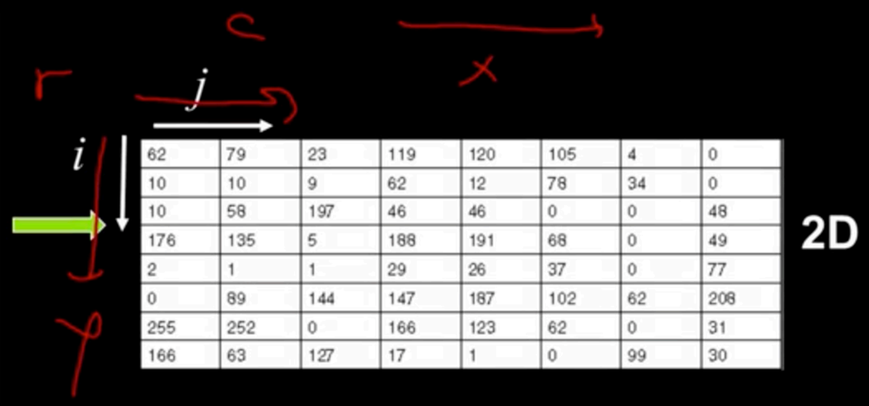
2D

A color image is just three functions "stacked" together. We can write this as a "vector-valued" function:

$$f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$$

Images as functions

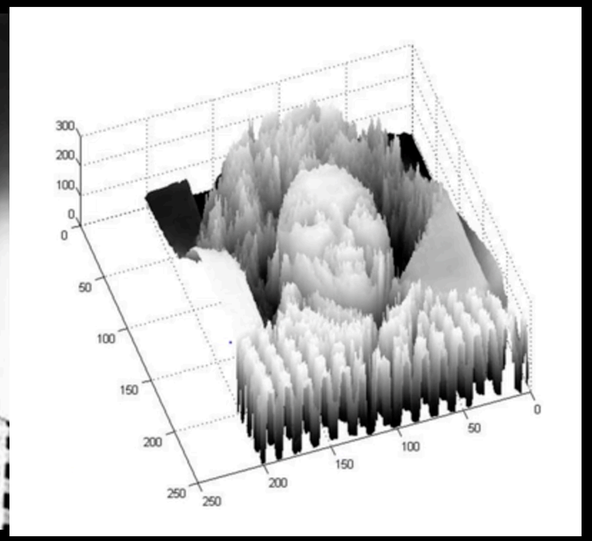


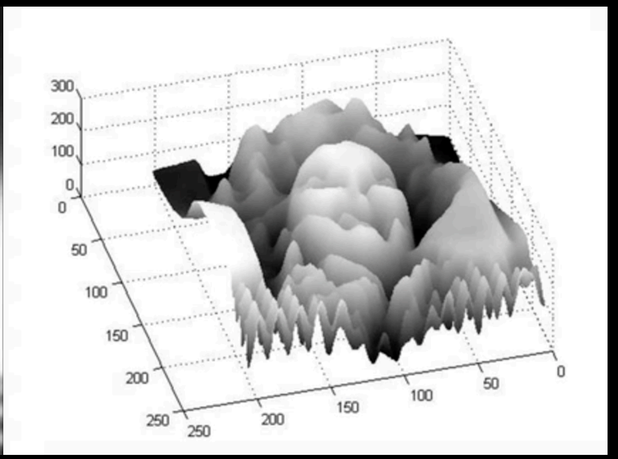


62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

2D

1D

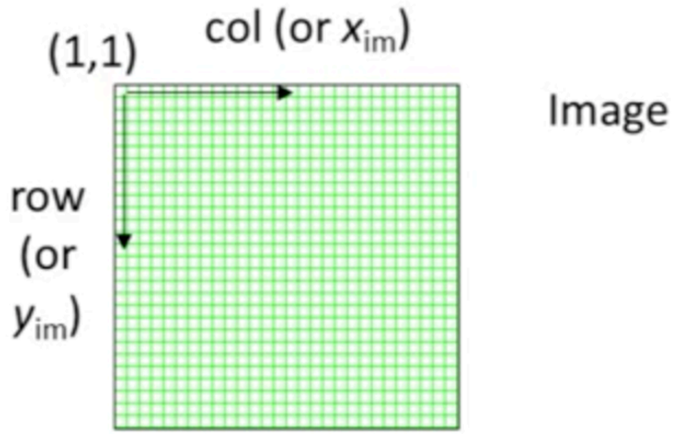




Images as functions

- Image can be represented by an *xy* plane
- Sampling partitions the plane into a grid of pixels, whose indices are integers
- We can also think of it as an *MxN* matrix of numbers
- We can index an element either by (row,col) or (*x*,*y*)

- We will use the convention
  - Top left pixel is (1,1)
  - *x* index (or column) increases to the right
  - *y* index (or row) increases down



Example

- An image has 8 bits per pixel
  - If unsigned, range of values is?
  - If signed (ie., two's complement), range is?
  - Number of bytes in a 3872 x 2592 pixel image?

Example (2)

N bits per pixel allows 2^N values

- An image has 8 bits per pixel
  - If unsigned, range of values is?

$0 .. (2^N - 1) = 0 .. (2^8 - 1) = 0..255$

  - If signed (ie., two's complement), range is?

$-(2^{N-1}) .. +(2^{N-1} - 1) \rightarrow -128 .. +127$

  - Number of bytes in a 3872 x 2592 pixel image?

$3872 \times 2592 = 10,036,224$

10 MB

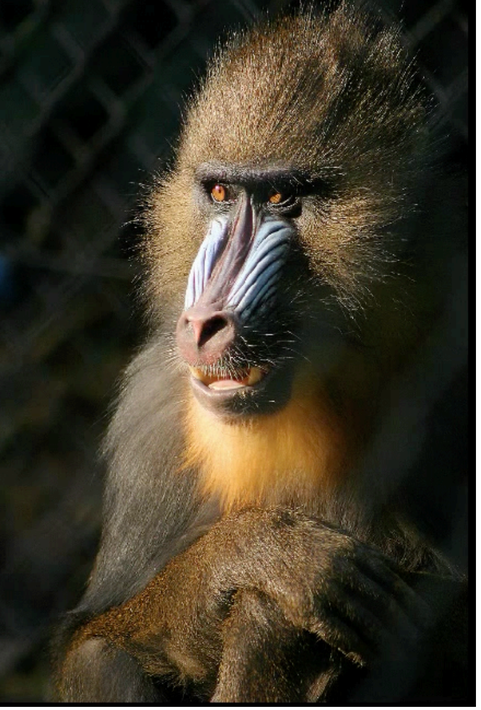


Image size, image resolution and spatial resolution

Image size



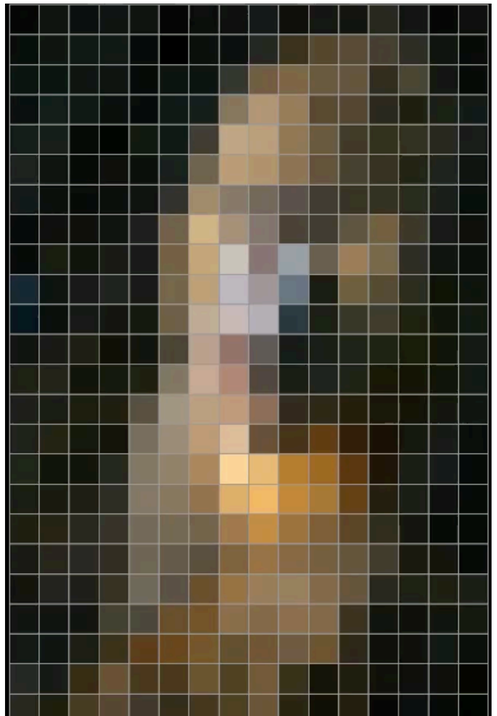
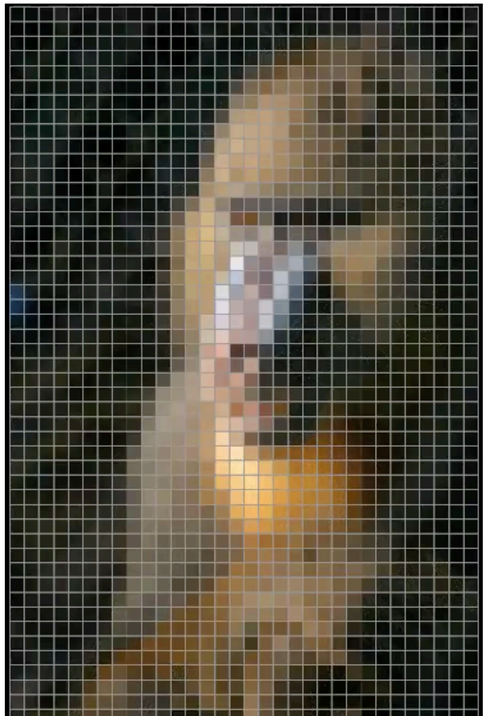
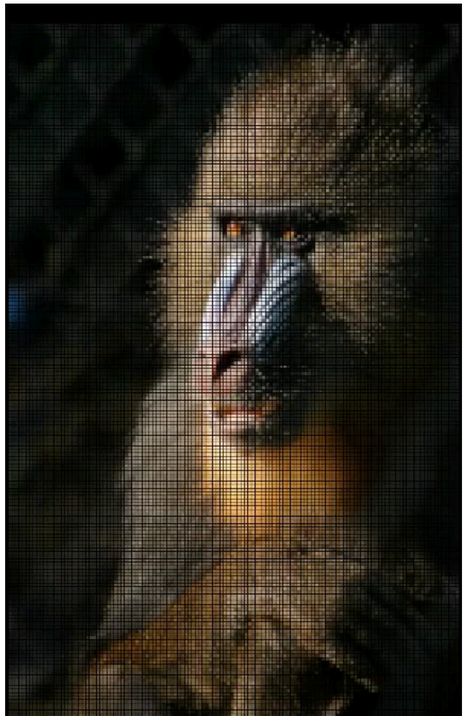
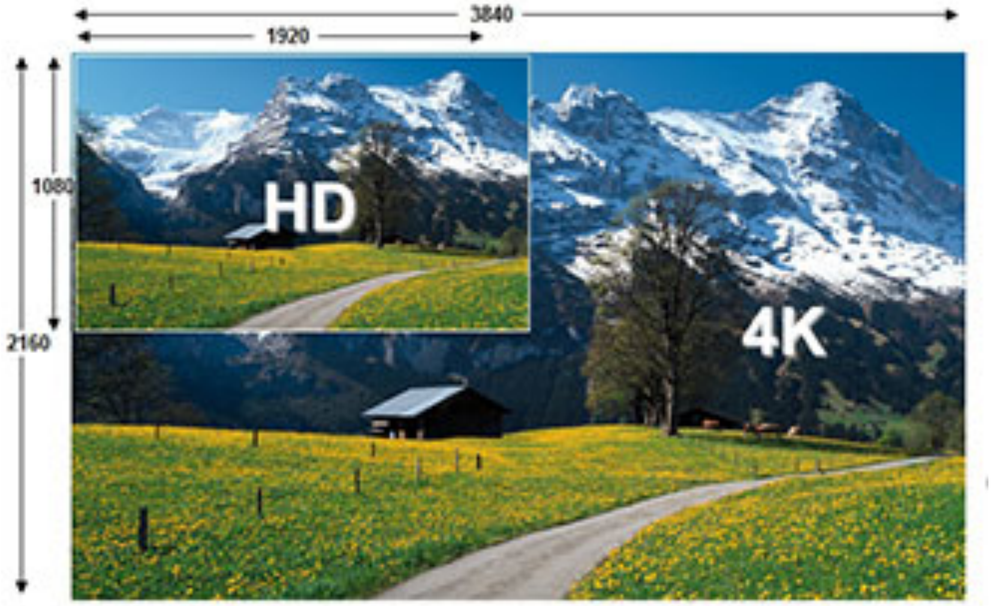
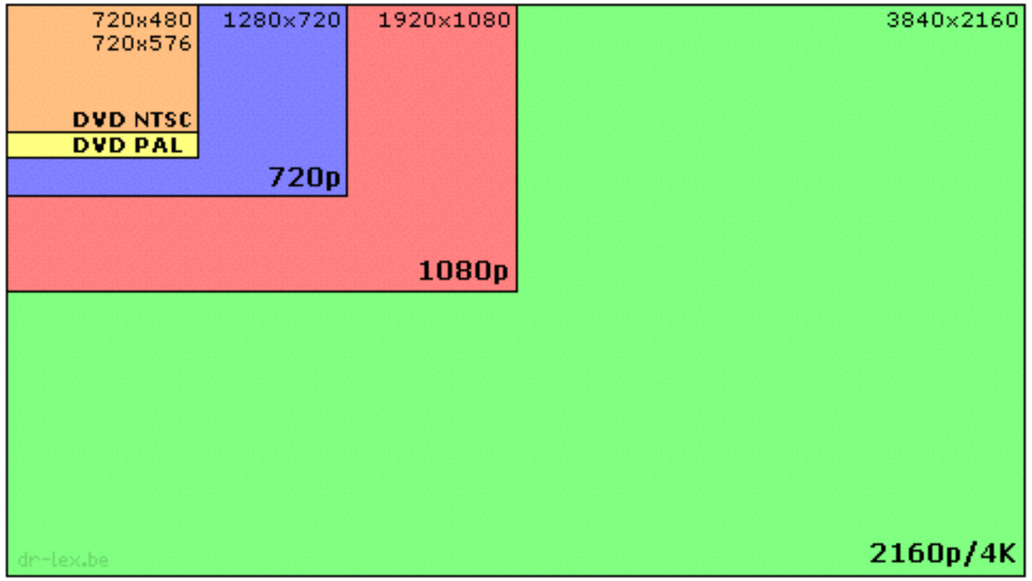




Image Size

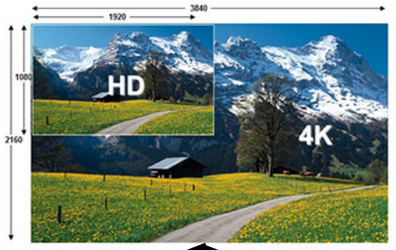




 # pixels (width x height)

Image Resolution

Image



Display

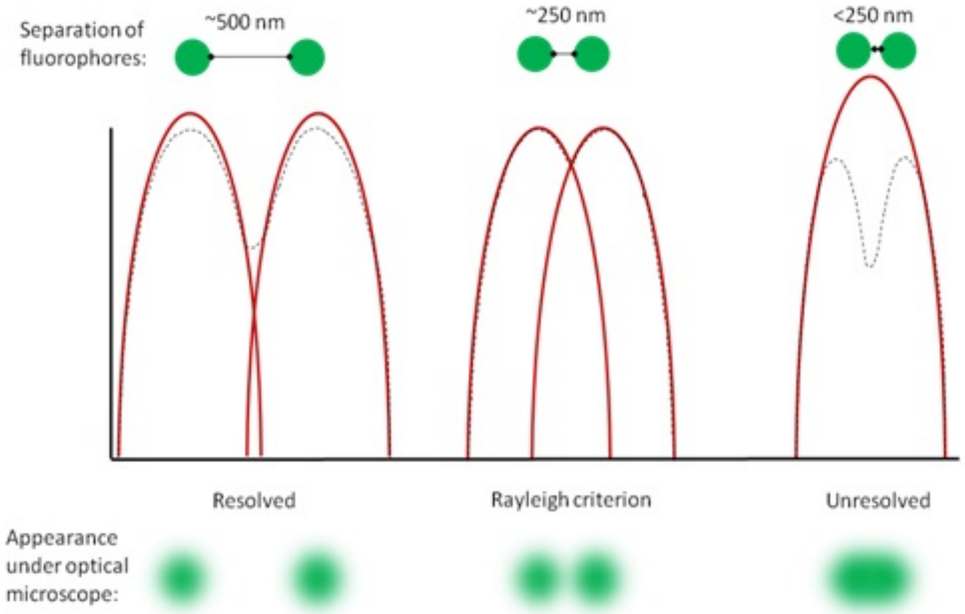
TV

Laptop

Phone

DPI

Spatial Resolution



- To resolve an object, we must use a wavelength equal to the size of the object or smaller

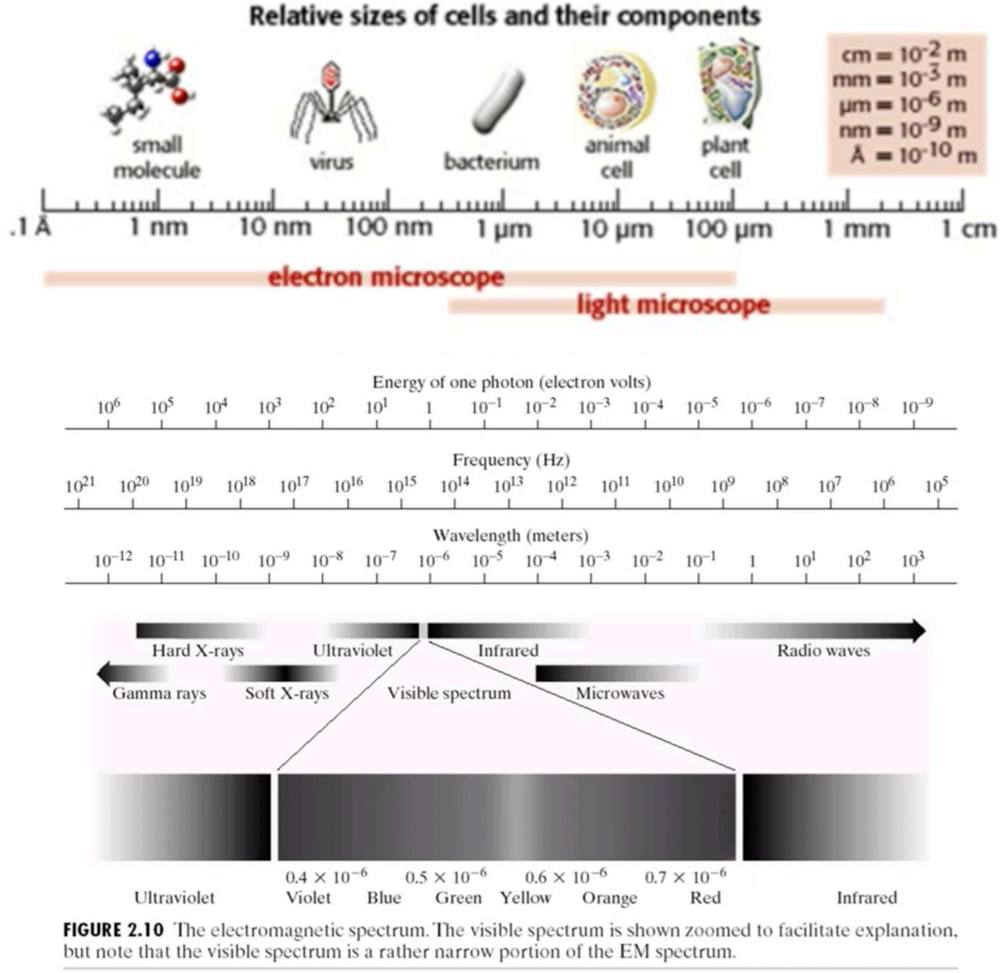
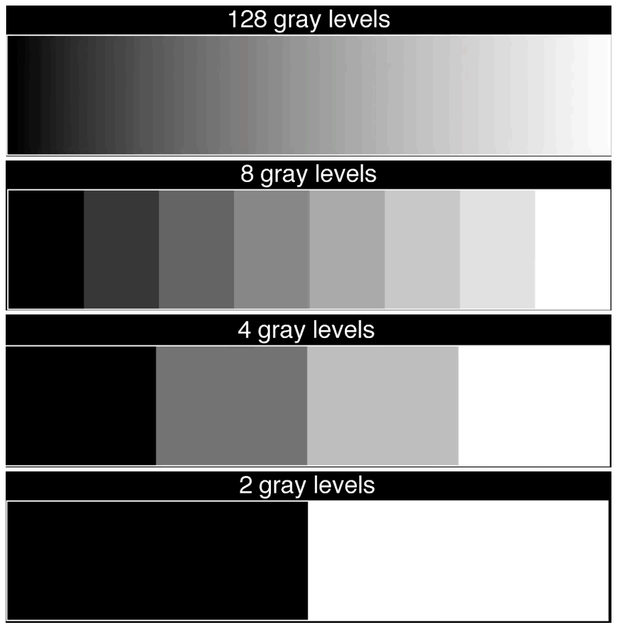


FIGURE 2.10 The electromagnetic spectrum. The visible spectrum is shown zoomed to facilitate explanation, but note that the visible spectrum is a rather narrow portion of the EM spectrum.

Color levels and Intensity Resolution

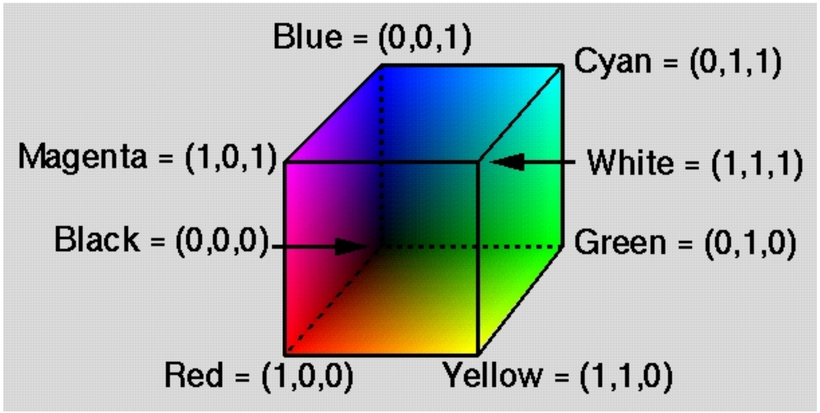


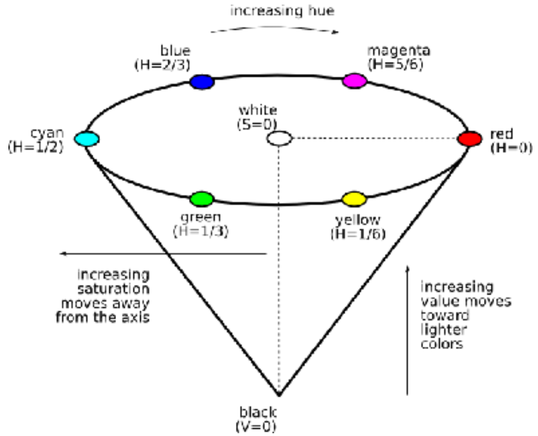
Summary

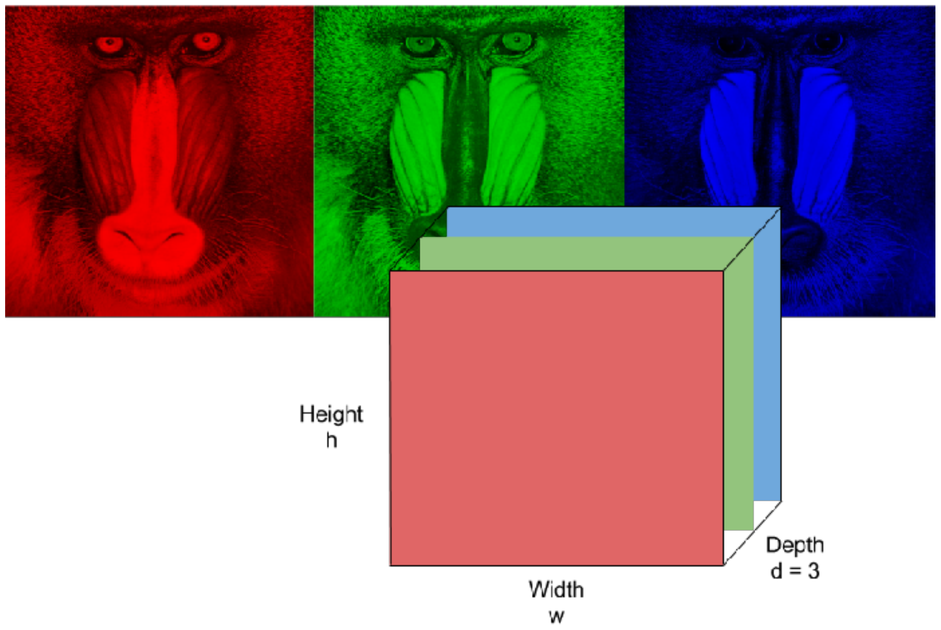
- image size (# pixels / pixel size) influences image resolution
- # color / gray levels influences intensity resolution
- Spatial / Intensity resolution intended to quantify the smallest details that can be perceived or recognized in a digital image.
- Spatial resolution = smallest change in position that can be recognized / *smallest distance between physical points that the imaging system can resolve*.
- Intensity resolution = smallest change in brightness that can be recognized.

$$f(x,y) = \begin{bmatrix} r(x,y) \\ g(x,y) \\ b(x,y) \end{bmatrix}$$

Graylevel vs. Color Images







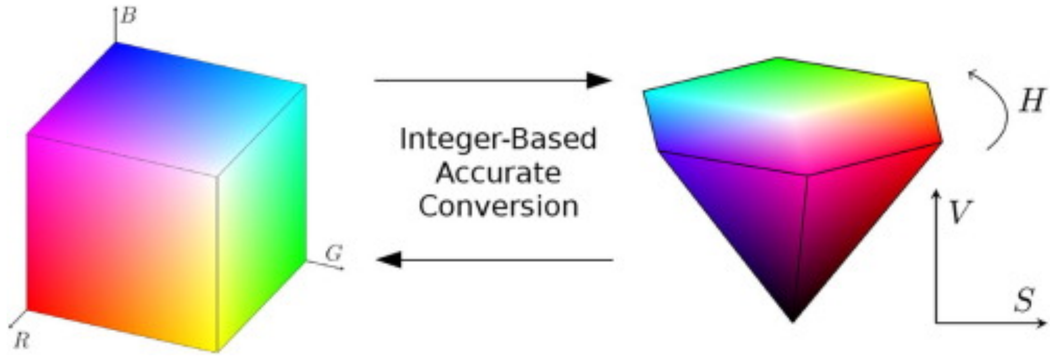
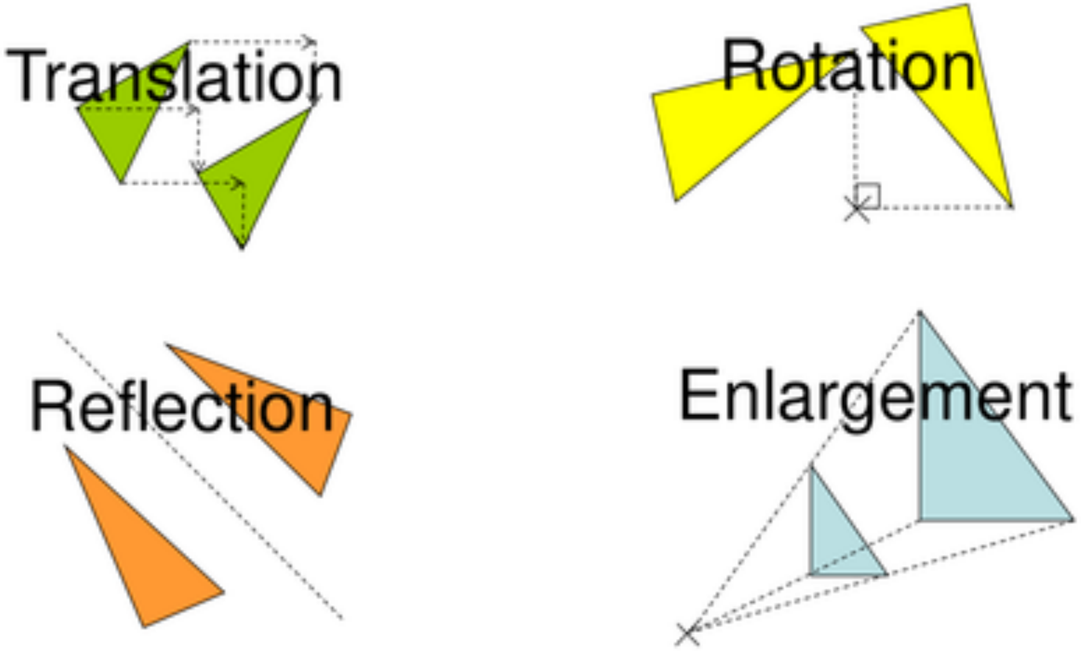
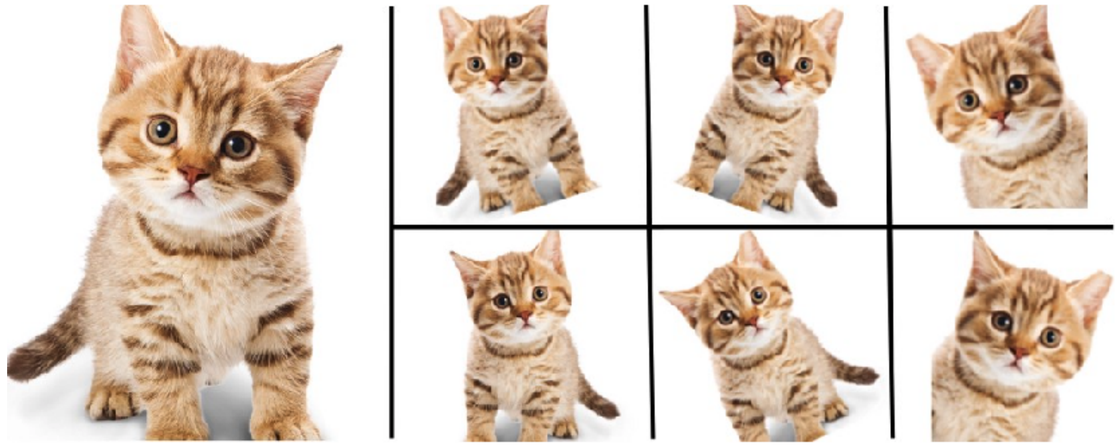


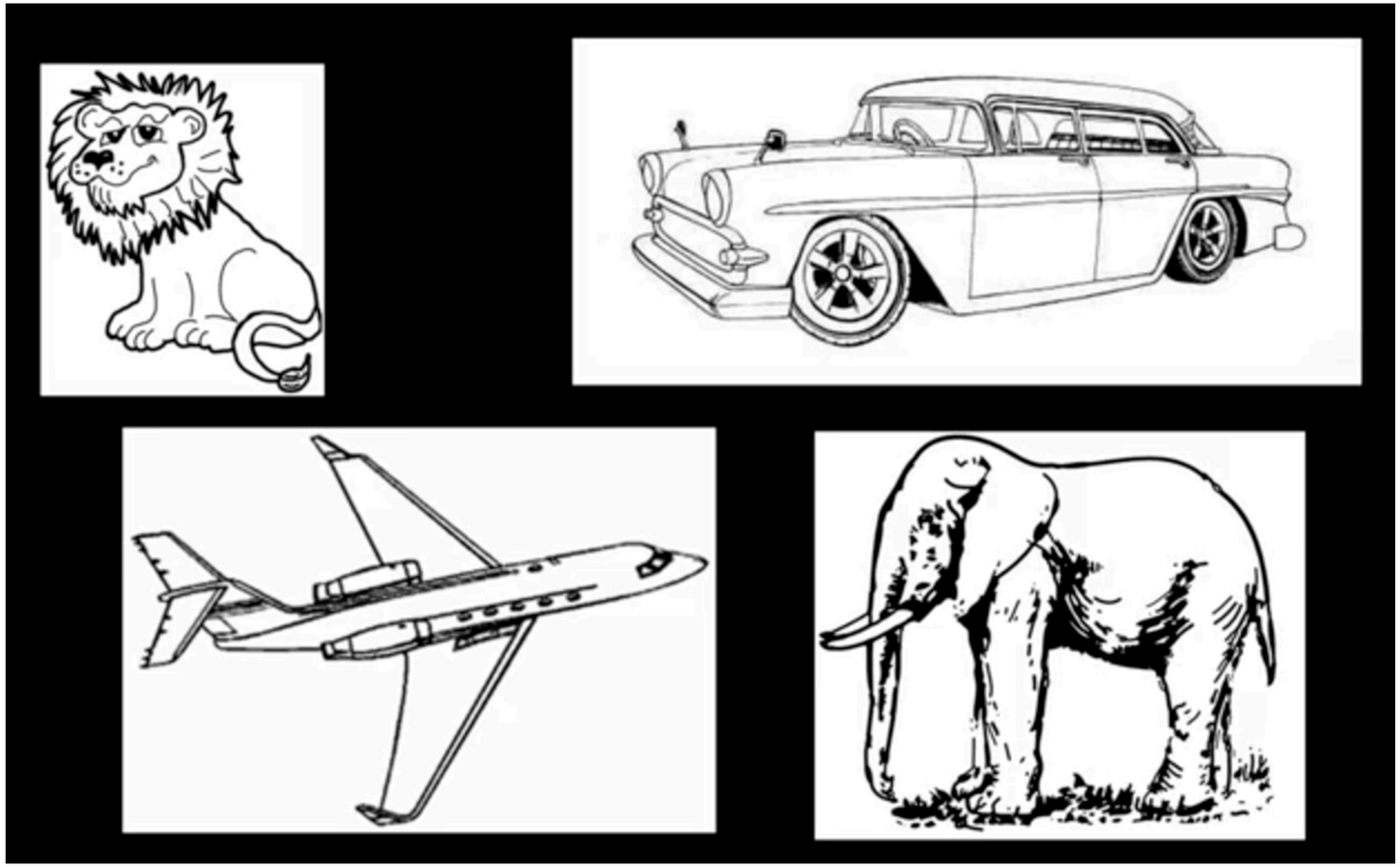
Image Transformations





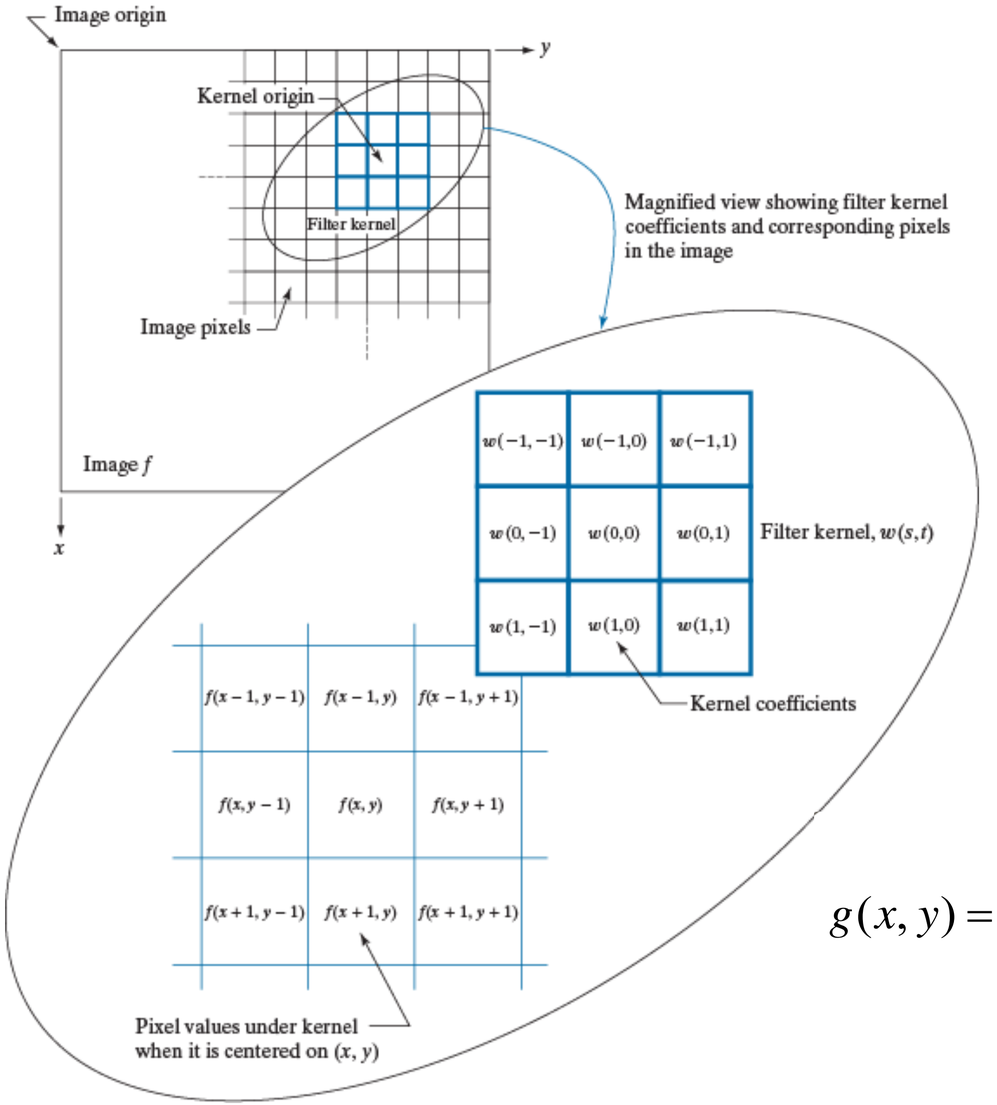
Enlarge your Dataset

Spatial filtering++



«Reduced images» - Edges seem to be important

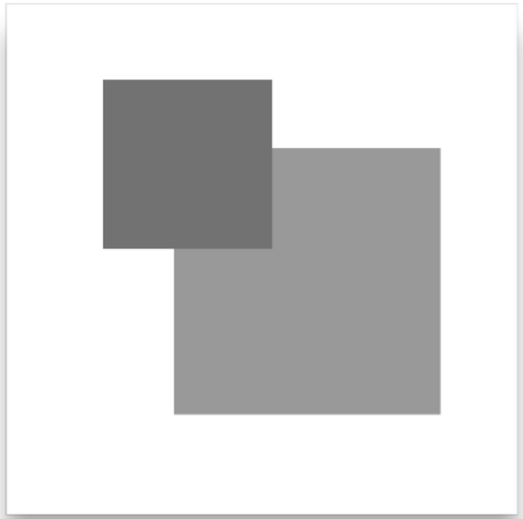
Spatial filtering++

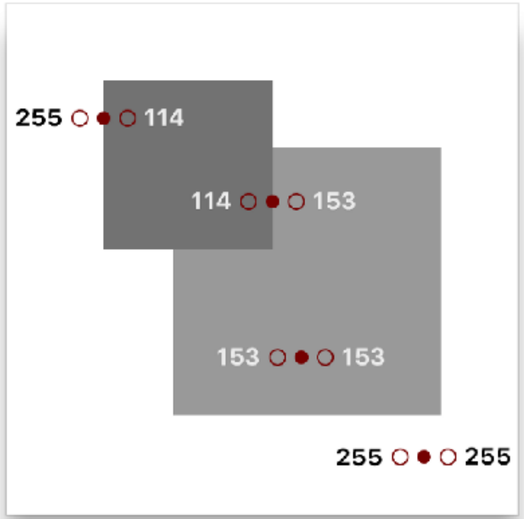


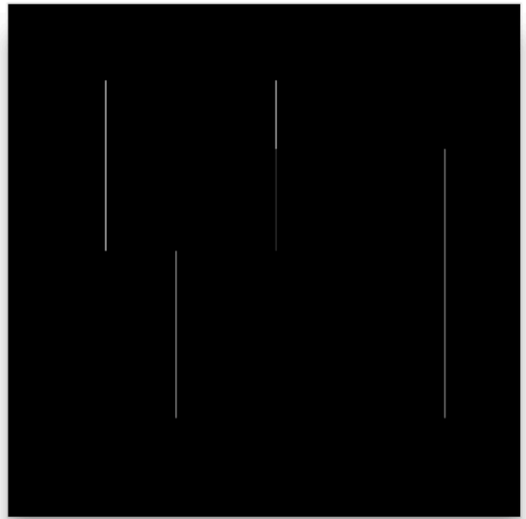
- Neighborhood processing / filtering:
  - Smoothing
  - **Sharpening**

$$g(x,y) = \sum_{s=-m/2}^{m/2} \sum_{t=-n/2}^{n/2} w(s,t)\, f(x+s, y+t)$$

Sharpening - Finite Differences







```
kernel = [1  0 -1]
```

```
kernel = [
  [0  0  0]
  [1  0 -1]
  [0  0  0]
]
```

255 × 0	255 × 0	255 × 0	114	114	114	114
255 × 1	255 × 0	255 × -1	114	114	114	114
255 × 0	255 × 0	255 × 0	114	114	114	114
255	255	255	114	114	114	114
255	255	255	255	255	255	255
255	255	255	255	255	255	255
255	255	255	255	255	255	255

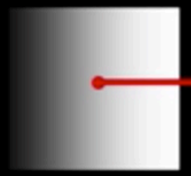
0				

Image gradient

The gradient of an image: $\nabla f = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}]$

The gradient direction is given by: $\theta = \tan^{-1}(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x})$

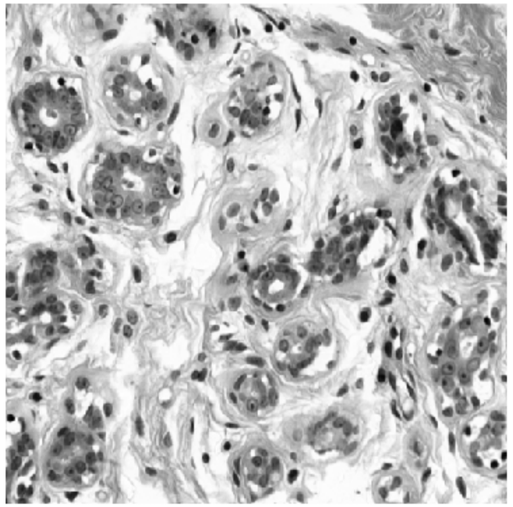
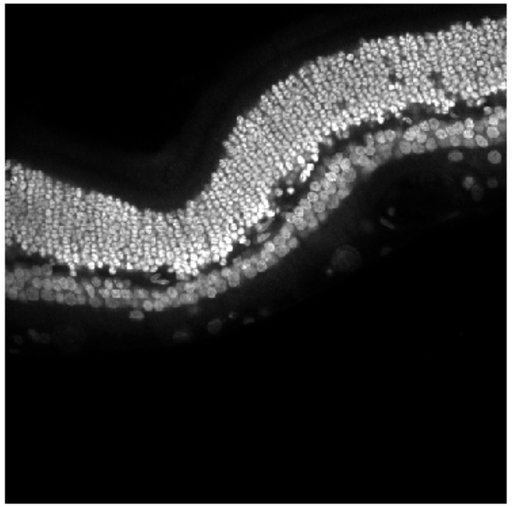
The *amount of change* is given by the gradient magnitude: $\|\nabla f\| = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2}$

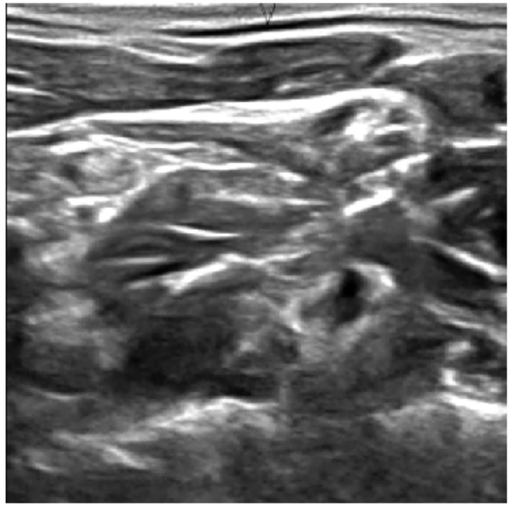

$\nabla f = [\frac{\partial f}{\partial x}, 0]$


$\nabla f = [0, \frac{\partial f}{\partial y}]$

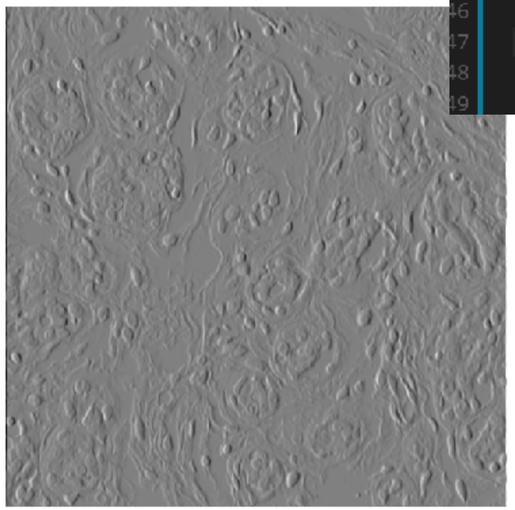
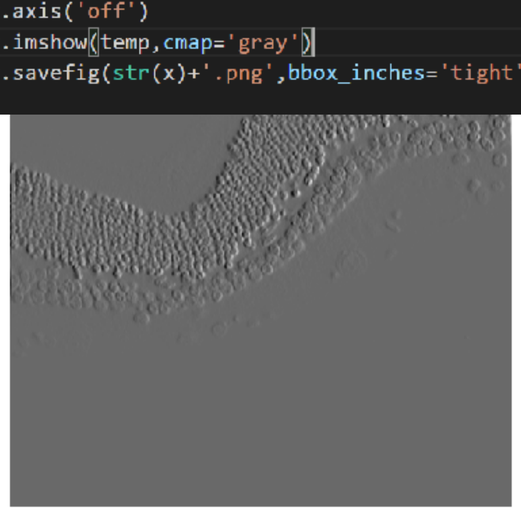
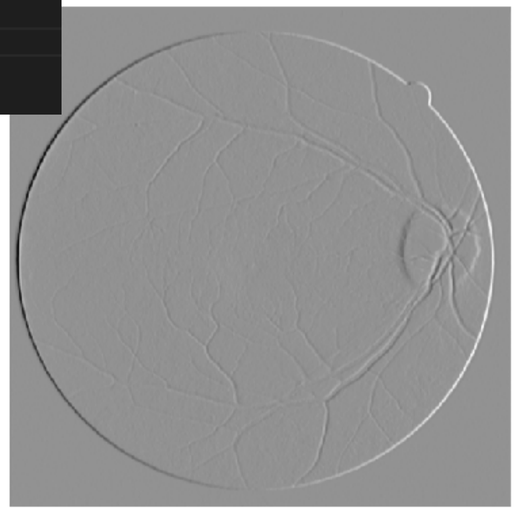
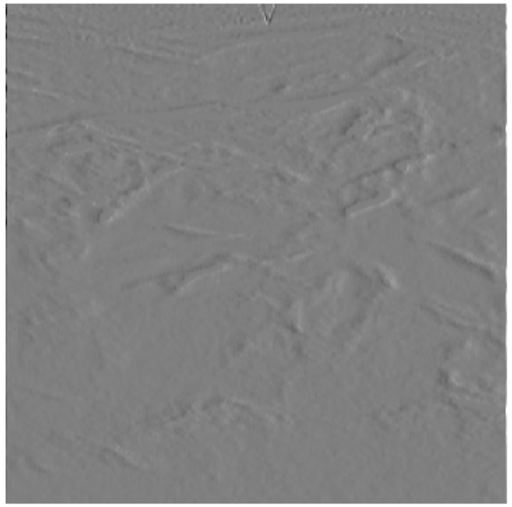
$\nabla f = [\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}]$

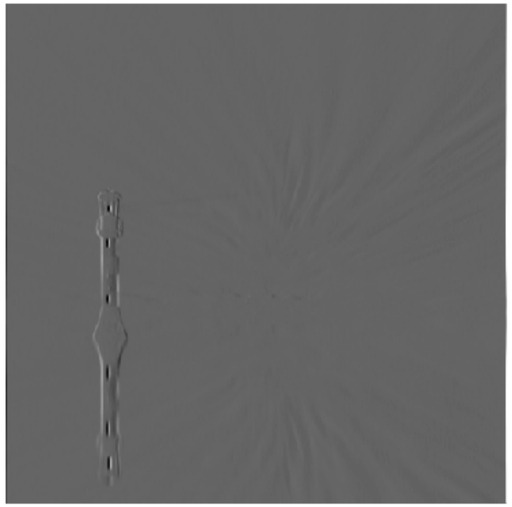
Sharpening…

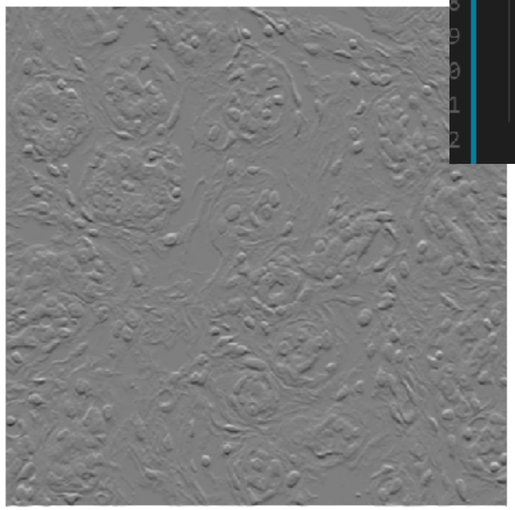
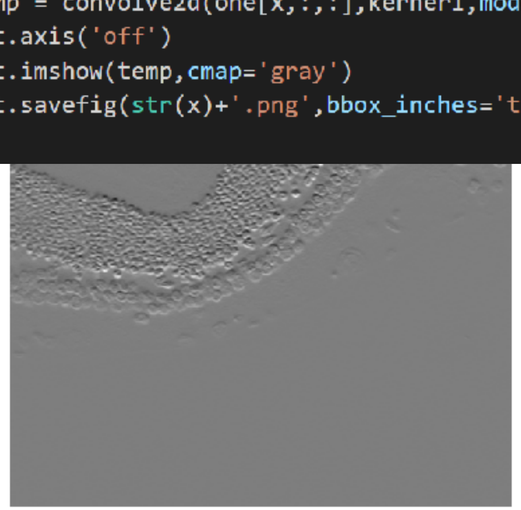
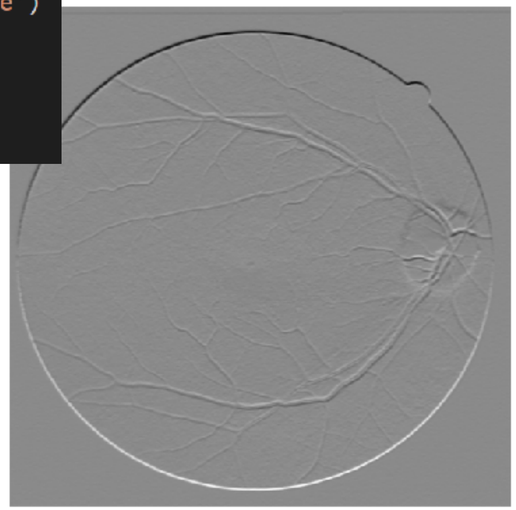
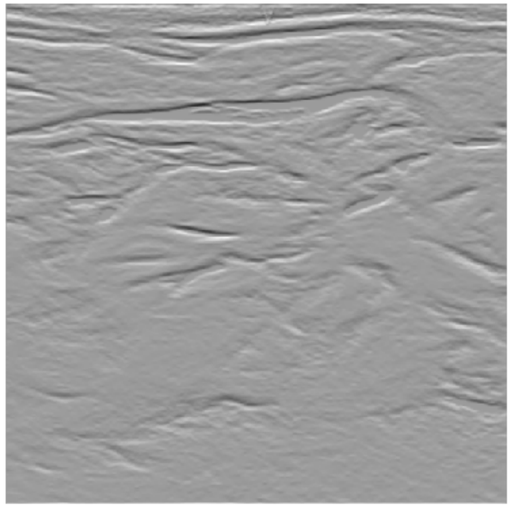


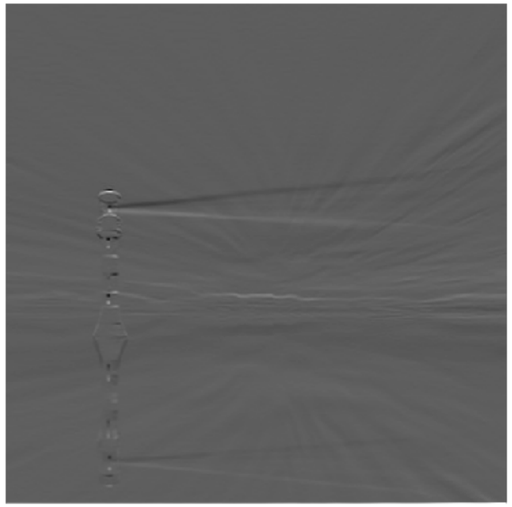
```python
# 2. Edge Detection (Horizontal)
kernerl = np.array([
    [-1,0,1],
    [-1,0,1],
    [-1,0,1]
])

for x in range(len(one)):
    temp = convolve2d(one[x,:,:],kernerl,mode='same')
    plt.axis('off')
    plt.imshow(temp,cmap='gray')
    plt.savefig(str(x)+'.png',bbox_inches='tight')
```



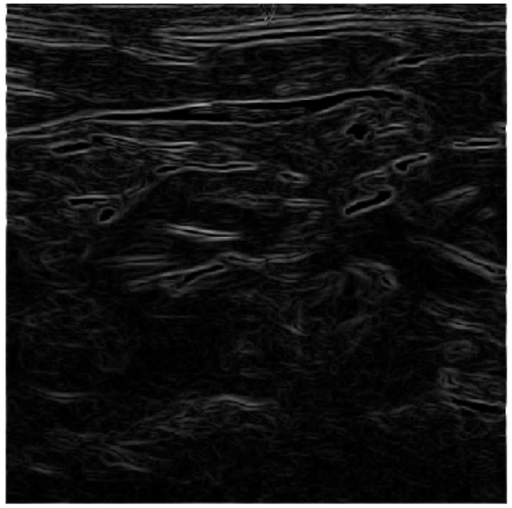
```python
# 3. Edge Detection (Vertical)
kernerl = np.array([
    [-1,-1,-1],
    [0,0,0],
    [1,1,1]
])

for x in range(len(one)):
    temp = convolve2d(one[x,:,:],kernerl,mode='same')
    plt.axis('off')
    plt.imshow(temp,cmap='gray')
    plt.savefig(str(x)+'.png',bbox_inches='tight')
```

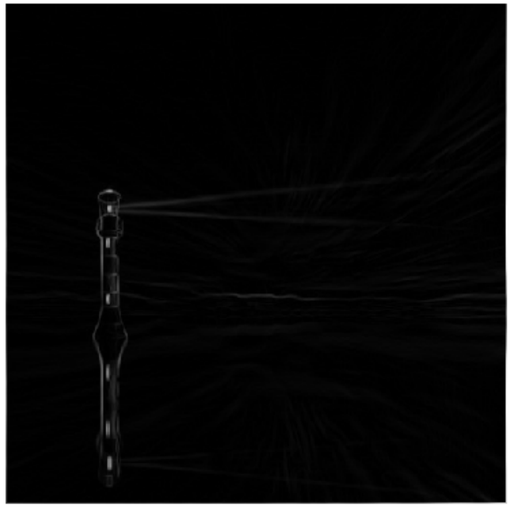







$$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$$

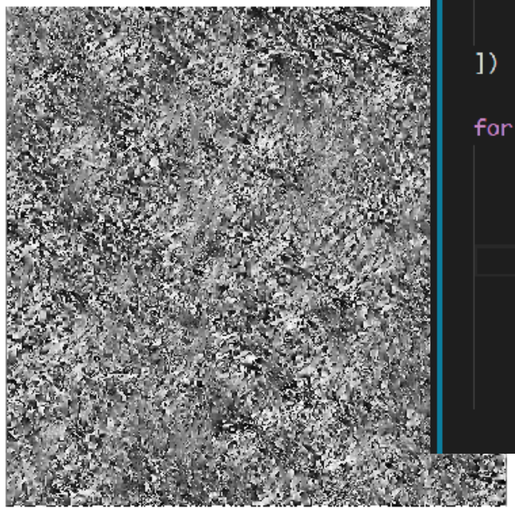
```python
# 4. Gradient Magnitude
kernel1 = np.array([
    [-1,-1,-1],
    [0,0,0],
    [1,1,1]
])
kernel2 = np.array([
    [-1,0,1],
    [-1,0,1],
    [-1,0,1]
])

for x in range(len(one)):
    temp1 = convolve2d(one[x,:,:],kernel1,mode='same')
    temp2 = convolve2d(one[x,:,:],kernel2,mode='same')

    temp3 = np.sqrt(temp1**2 + temp2**2)

    plt.axis('off')
    plt.imshow(temp3,cmap='gray')
    plt.savefig(str(x)+'.png',bbox_inches='tight')
```

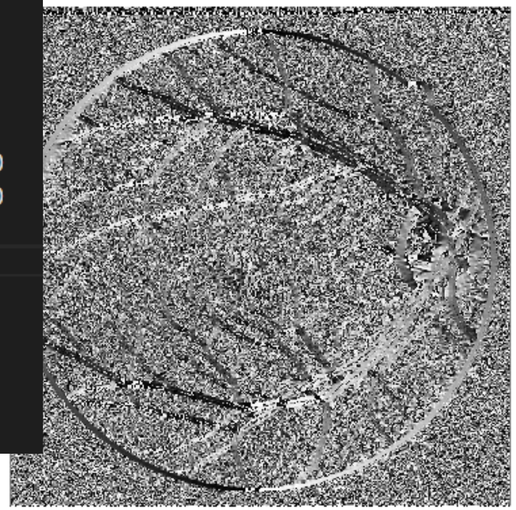


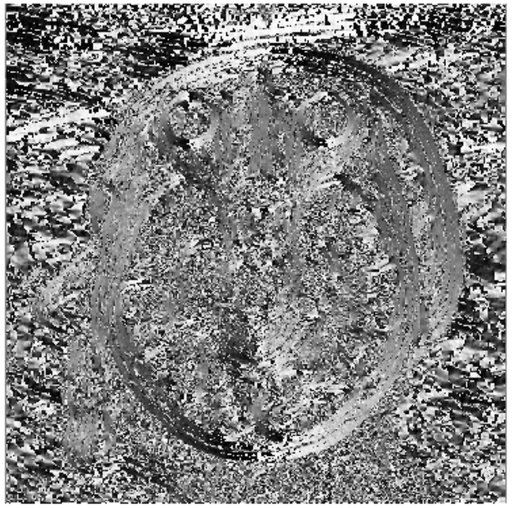
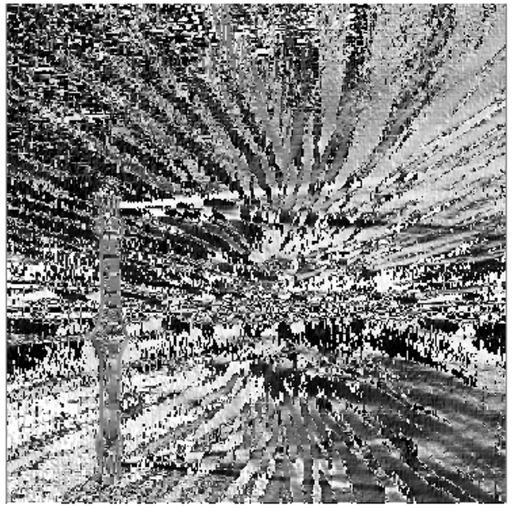


$$\Theta = \operatorname{atan}\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$$

```python
# 5. Gradient Direction
kernel1 = np.array([
    [-1,-1,-1],
    [0,0,0],
    [1,1,1]
])
kernel2 = np.array([
    [-1,0,1],
    [-1,0,1],
    [-1,0,1]
])

for x in range(len(one)):
    temp1 = convolve2d(one[x,:,:],kernel1,mode='same')
    temp2 = convolve2d(one[x,:,:],kernel2,mode='same')

    temp3 = np.arctan(temp1/temp2)

    plt.axis('off')
    plt.imshow(temp3,cmap='gray')
    plt.savefig(str(x)+'.png',bbox_inches='tight')
```

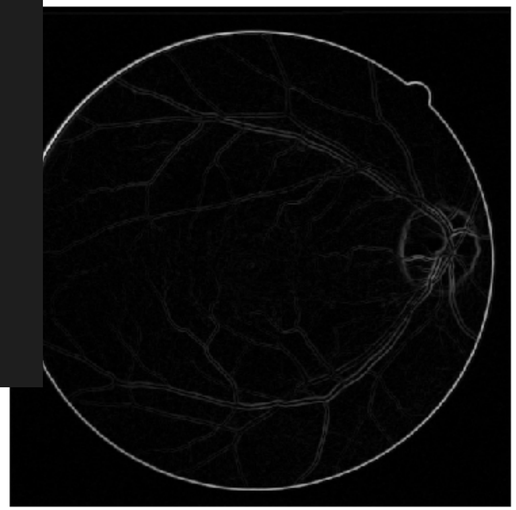




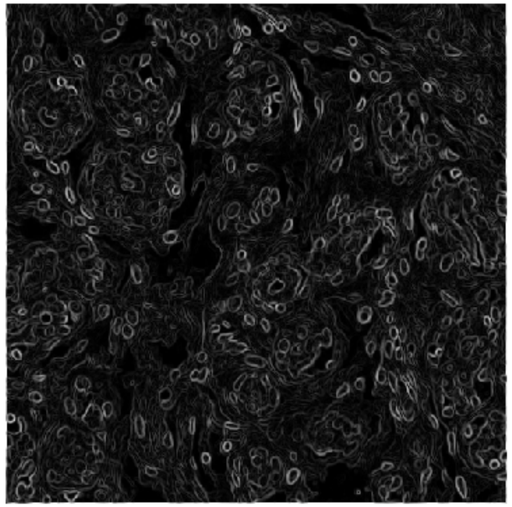



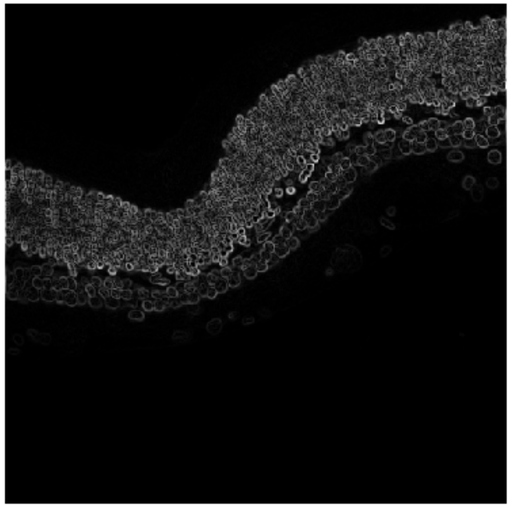
Sobel

```python
# 6. Sobel Gradient Magnitude
kernerl1 = np.array([
    [1,2,1],
    [0,0,0],
    [-1,-2,-1]
])
kernerl2 = np.array([
    [1,0,-1],
    [2,0,-2],
    [1,0,-1]
])

for x in range(len(one)):
    temp1 = convolve2d(one[x,:,:],kernerl1,mode='same')
    temp2 = convolve2d(one[x,:,:],kernerl2,mode='same')

    temp3 = np.sqrt(temp1**2 + temp2**2)

    plt.axis('off')
    plt.imshow(temp3,cmap='gray')
    plt.savefig(str(x)+'.png',bbox_inches='tight')
```



«Ordinary» Prewitt

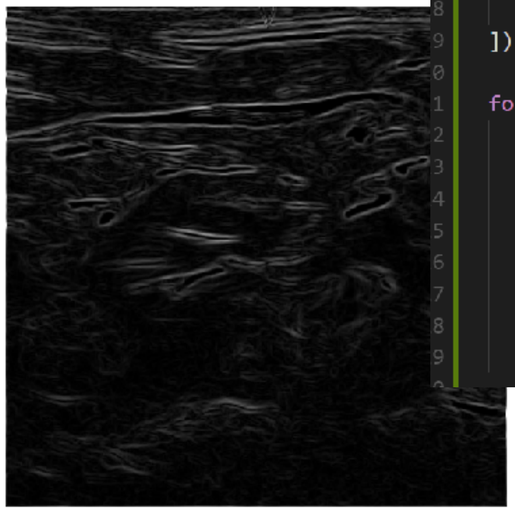


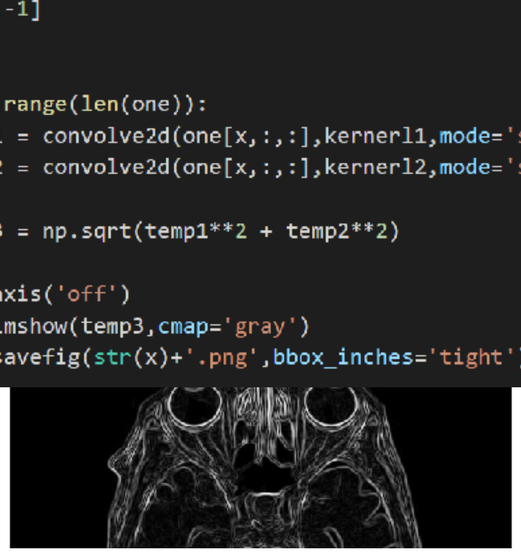


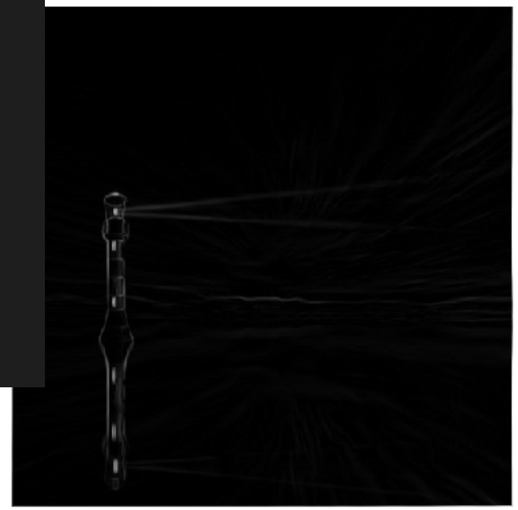
Sobel

Sobel

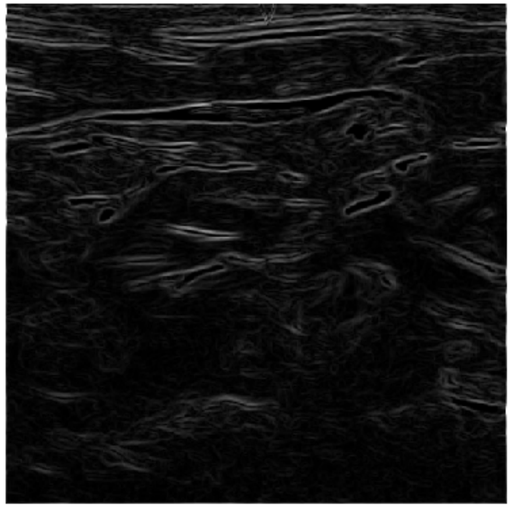
```
# 6. Sobel Gradient Magnitude
kernerl1 = np.array([
    [1,2,1],
    [0,0,0],
    [-1,-2,-1]
])
kernerl2 = np.array([
    [1,0,-1],
    [2,0,-2],
    [1,0,-1]
])

for x in range(len(one)):
    temp1 = convolve2d(one[x,:,:],kernerl1,mode='same')
    temp2 = convolve2d(one[x,:,:],kernerl2,mode='same')

    temp3 = np.sqrt(temp1**2 + temp2**2)

    plt.axis('off')
    plt.imshow(temp3,cmap='gray')
    plt.savefig(str(x)+'.png',bbox_inches='tight')
```

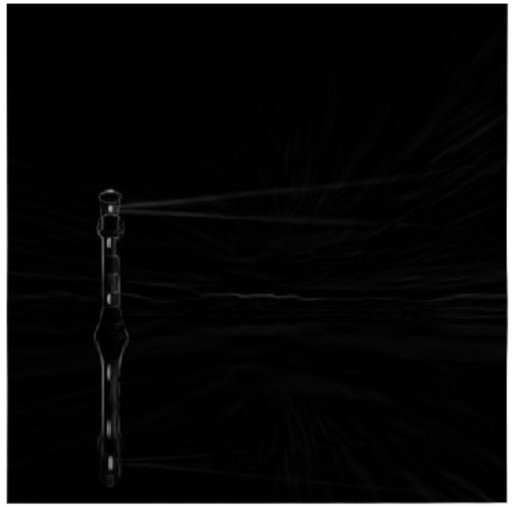






«Ordinary» Prewitt





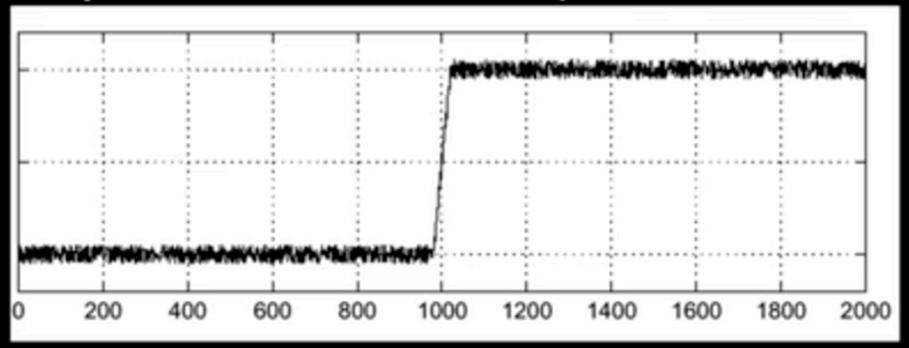
Sobel

Noise + derivative fiter = NotGood

Consider a single row or column of the image

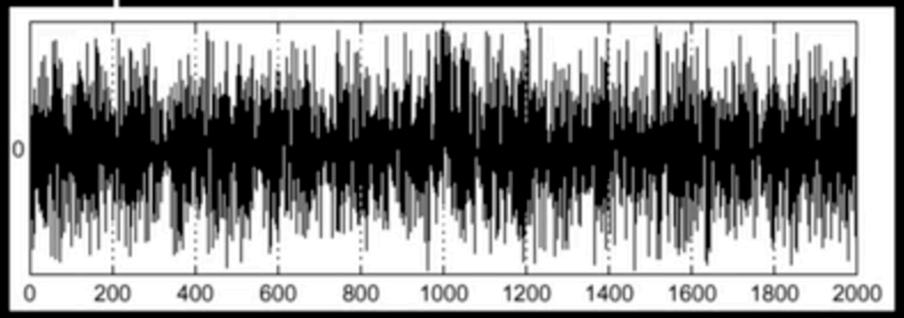
(plotting intensity as a function of *x)*

$f(x)$



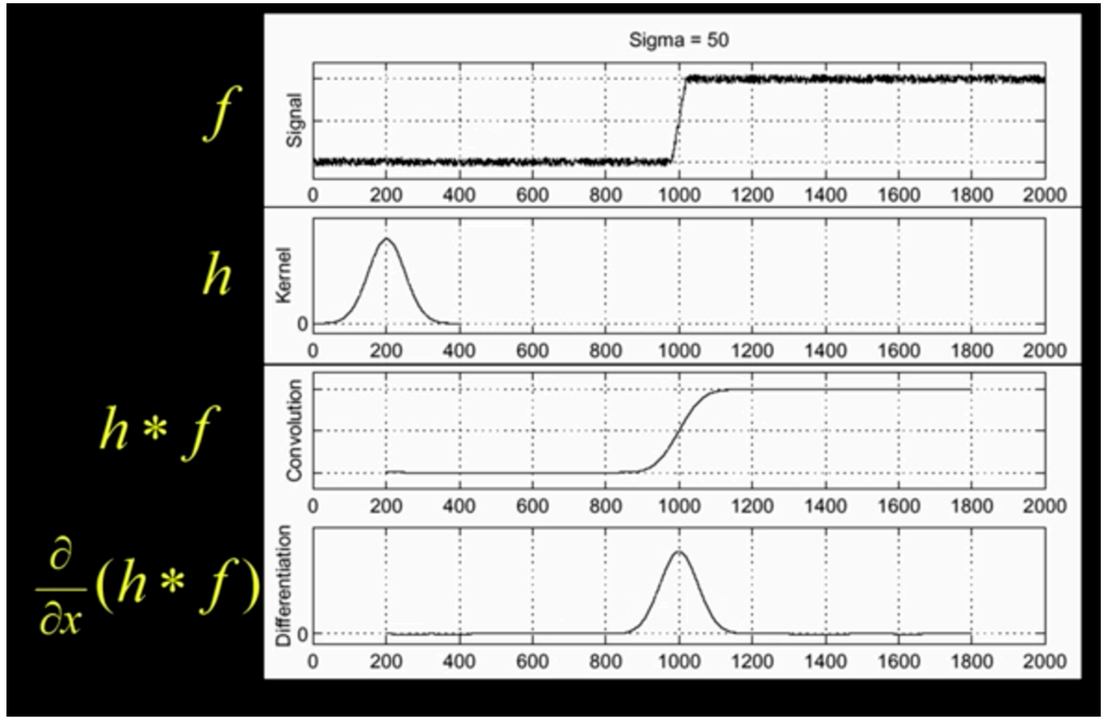
Apply derivative operator....

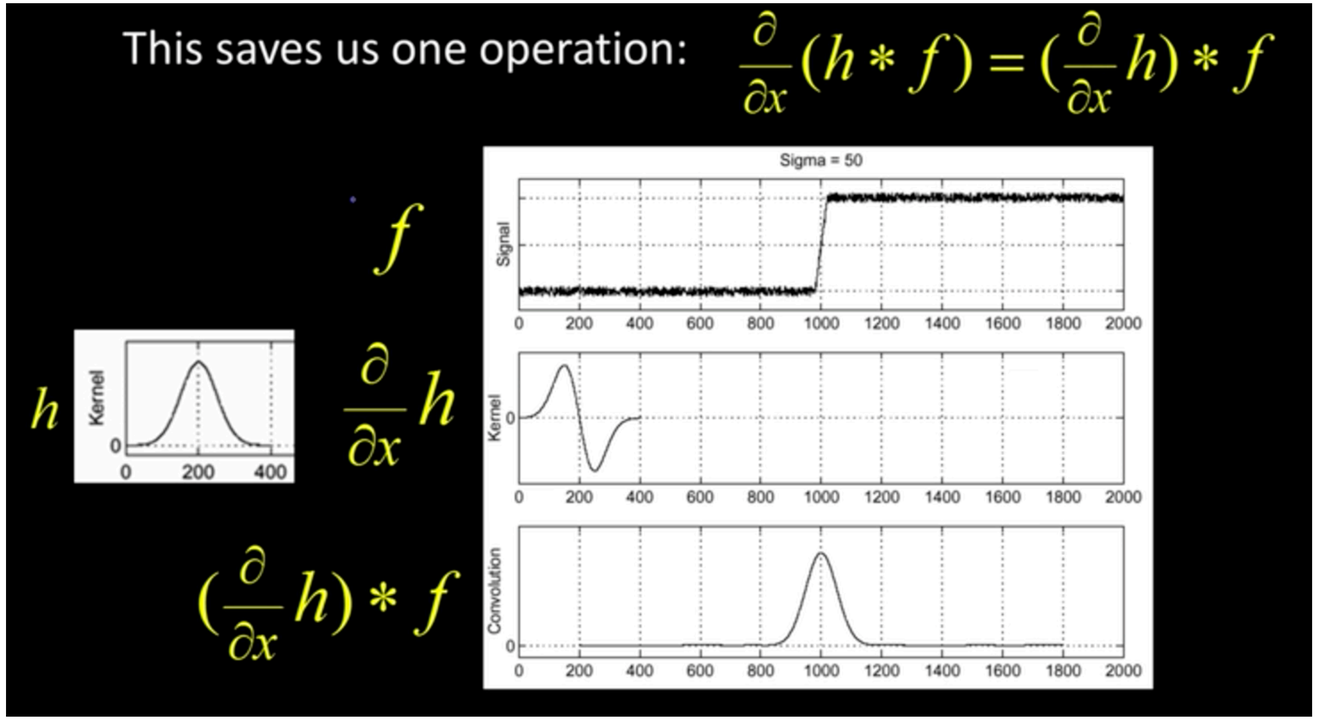
$\frac{d}{dx} f(x)$

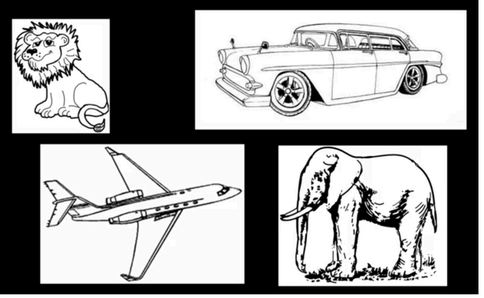


Uh, where's the edge?

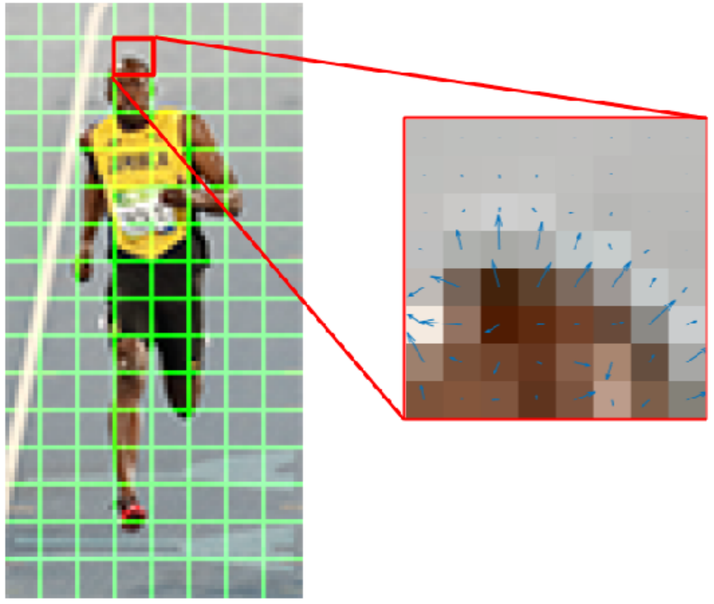
Smooth first / or combine







SotA before DL



2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

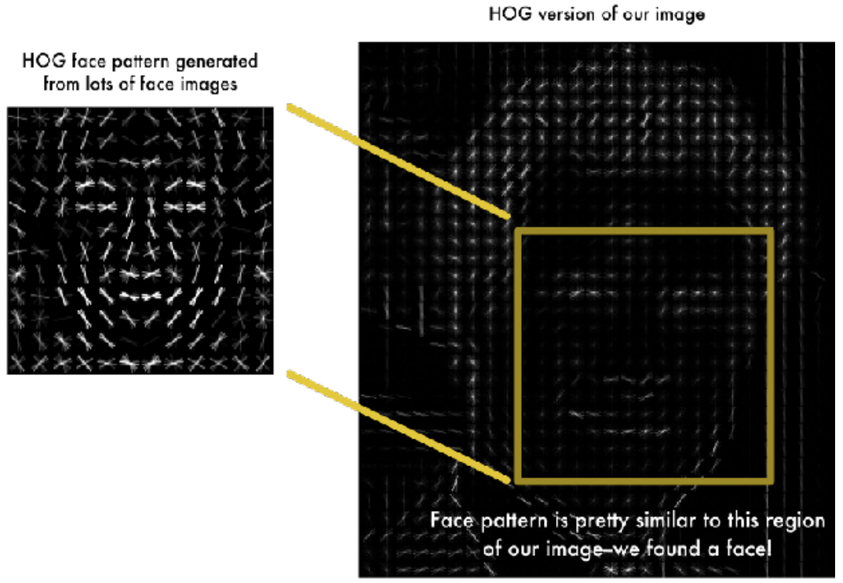
Gradient Magnitude

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

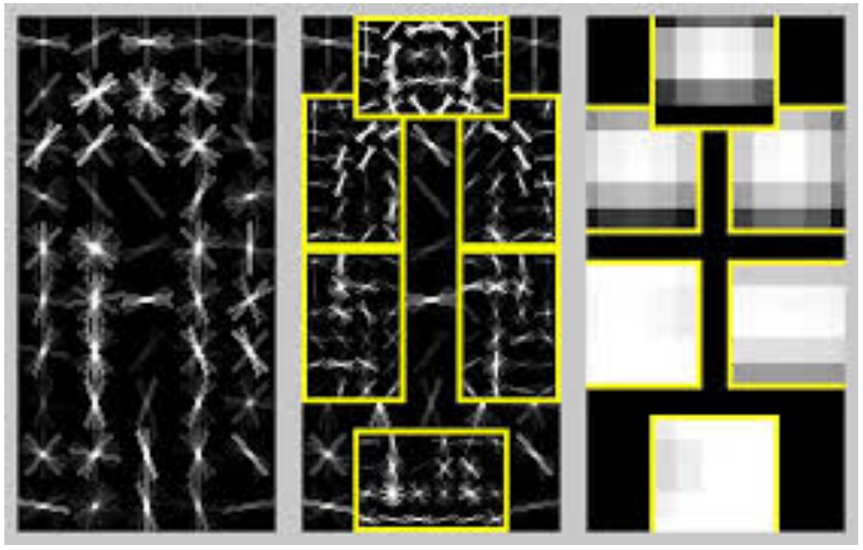
Gradient Direction

Gradient Magnitude & Direction

HOG



DPM



Feature engineering vs. learning