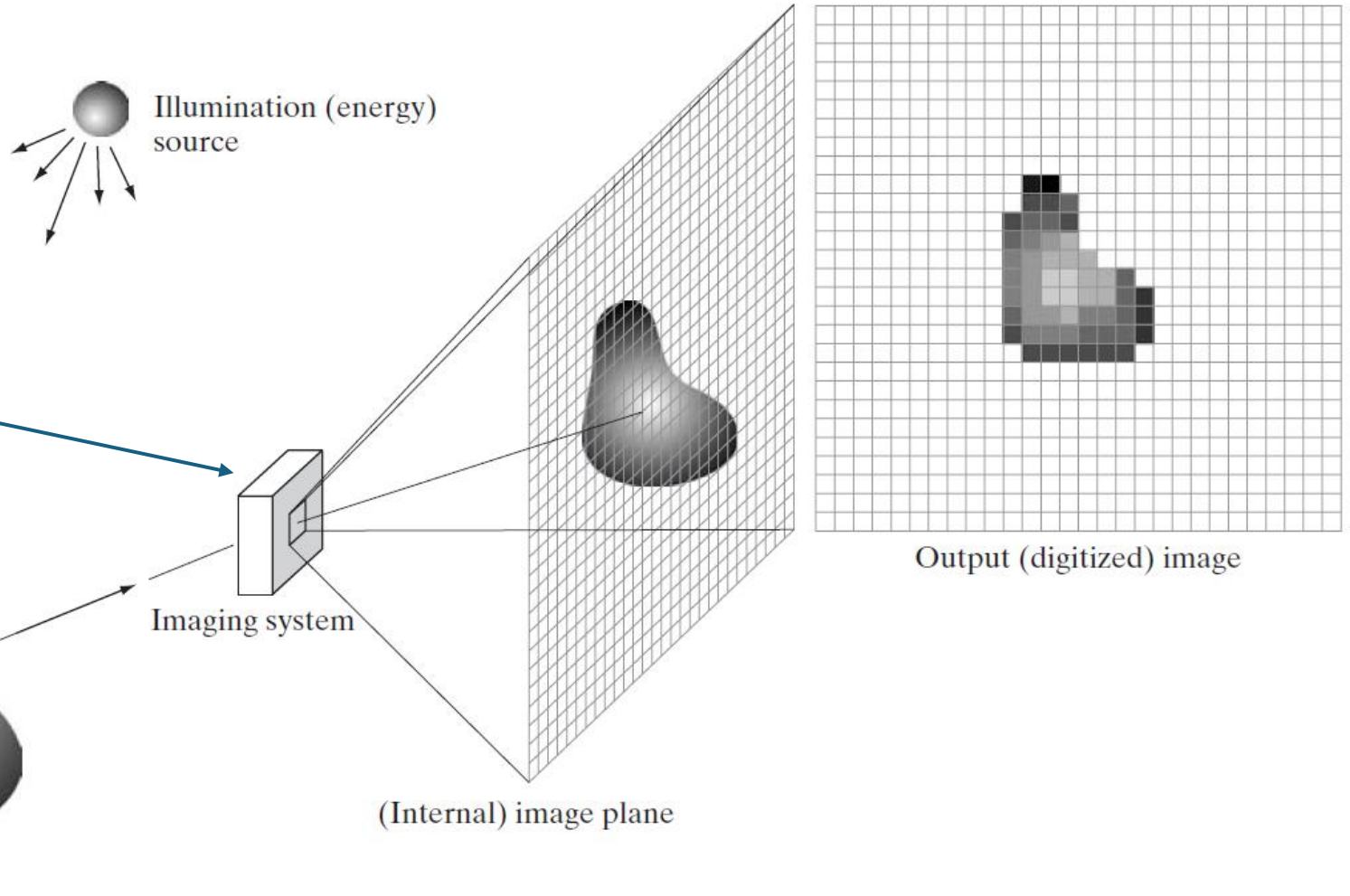
Digital Image Processing

Deep Learning and Image Processing

Digital Image Acquisition Process



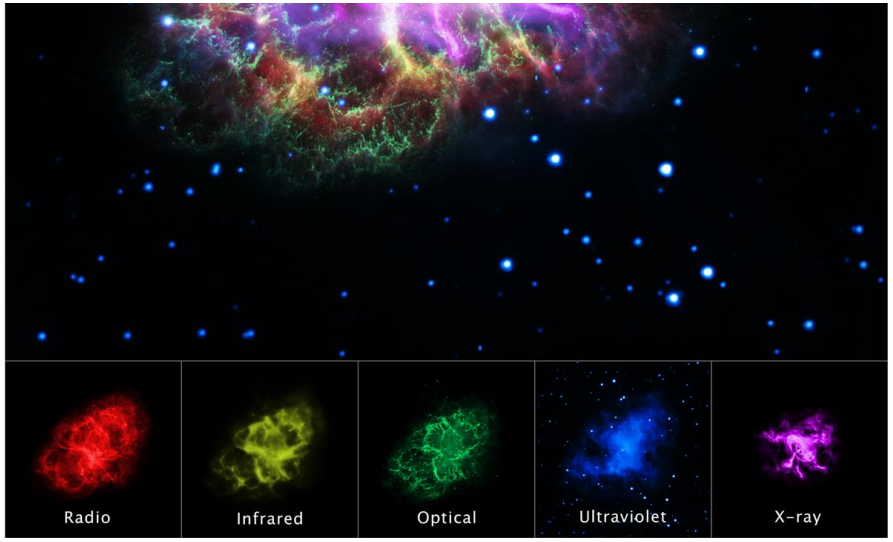
Energy Sources

Electromagnetic Spectrum

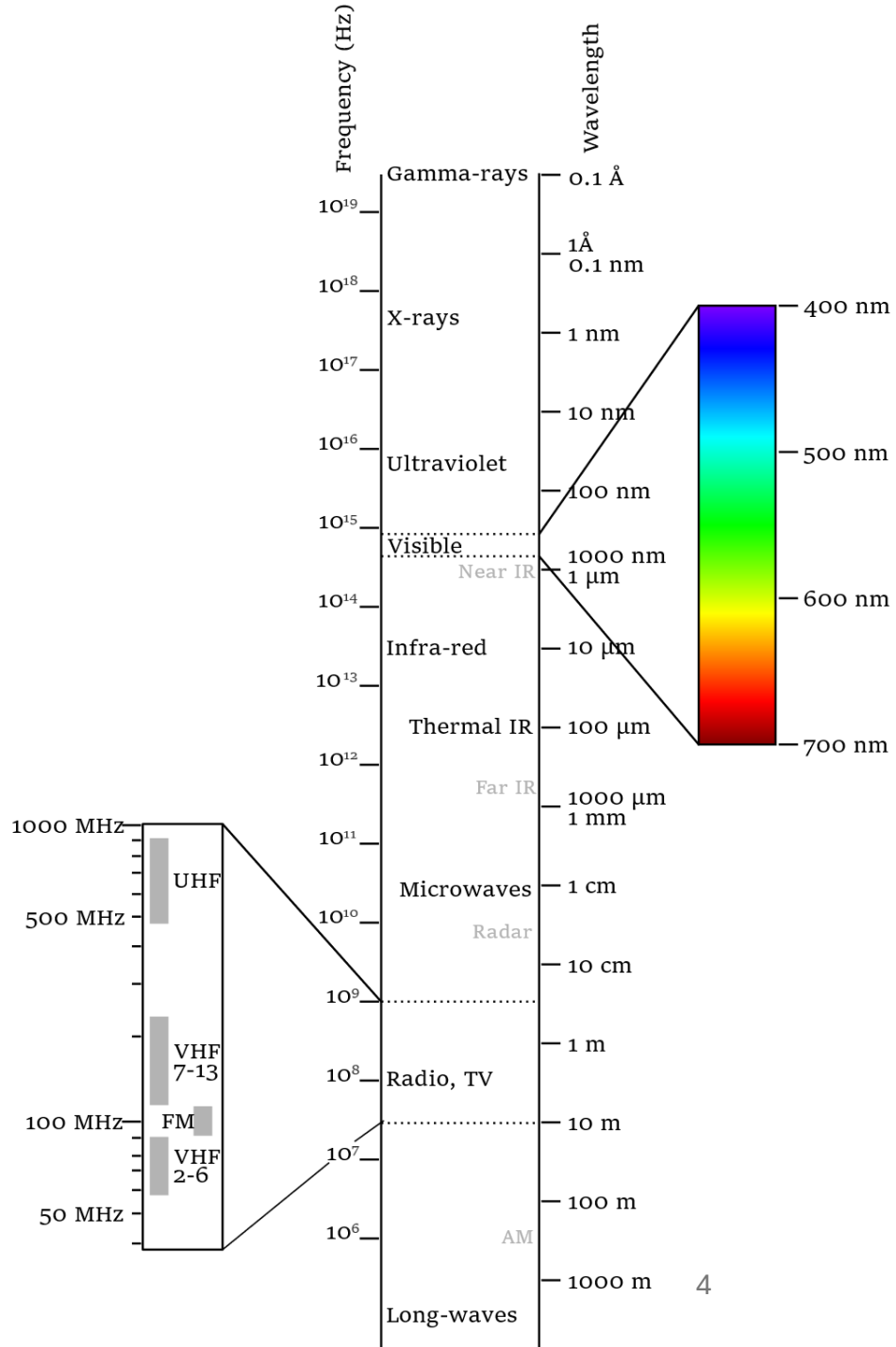
examples:

- visible band: digital camera
- microwaves: radar
- X-rays: computed tomography (CT)
- radio band: magnetic resonance imaging (MRI)
- gamma rays: positron emission tomography (PET)

Crab Nebula in multiple wavelengths



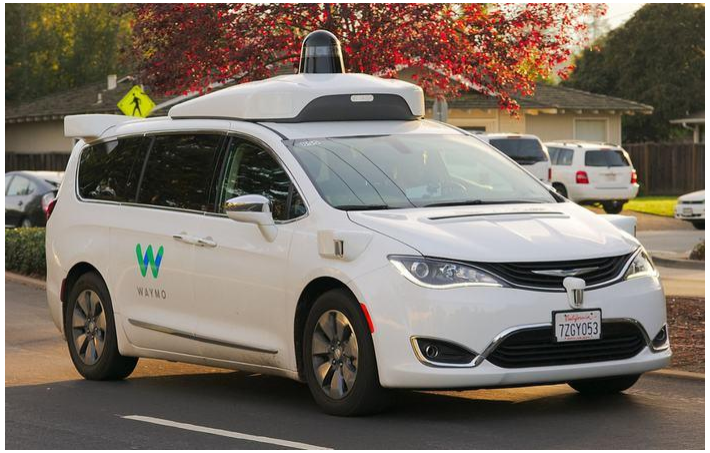
source



Lidar vs Cameras for Self-Driving Cars

Lidar:

- similar to radar, but using lasers
- very accurate



camera:

- human-like perception
- cost-effective

TESLA

Examples for other Energy Sources

ultrasound

electron beams
(electron microscopy)

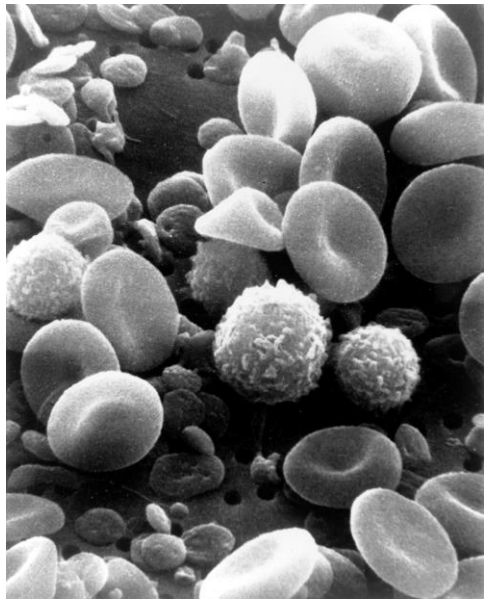
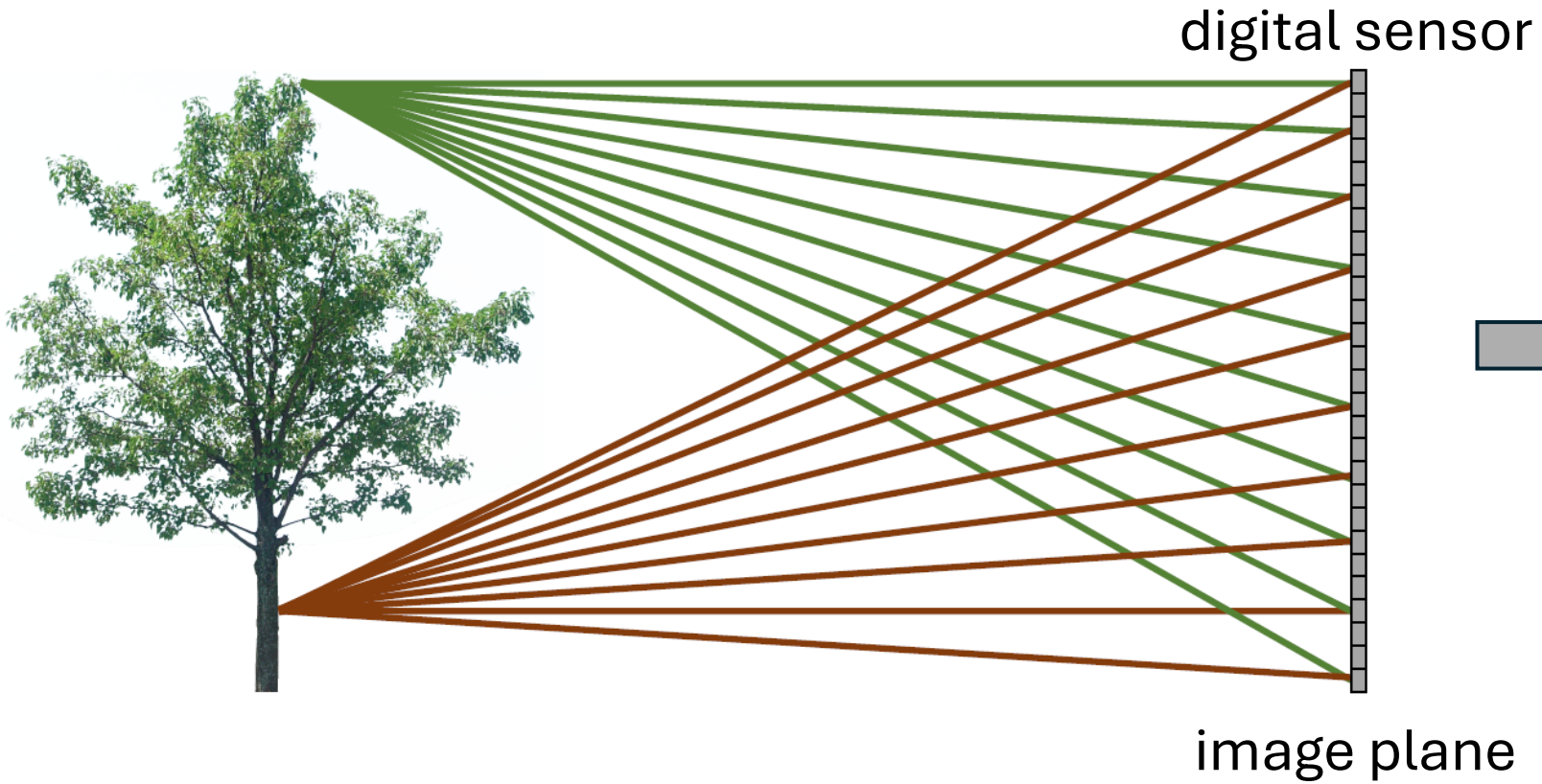


Image Formation

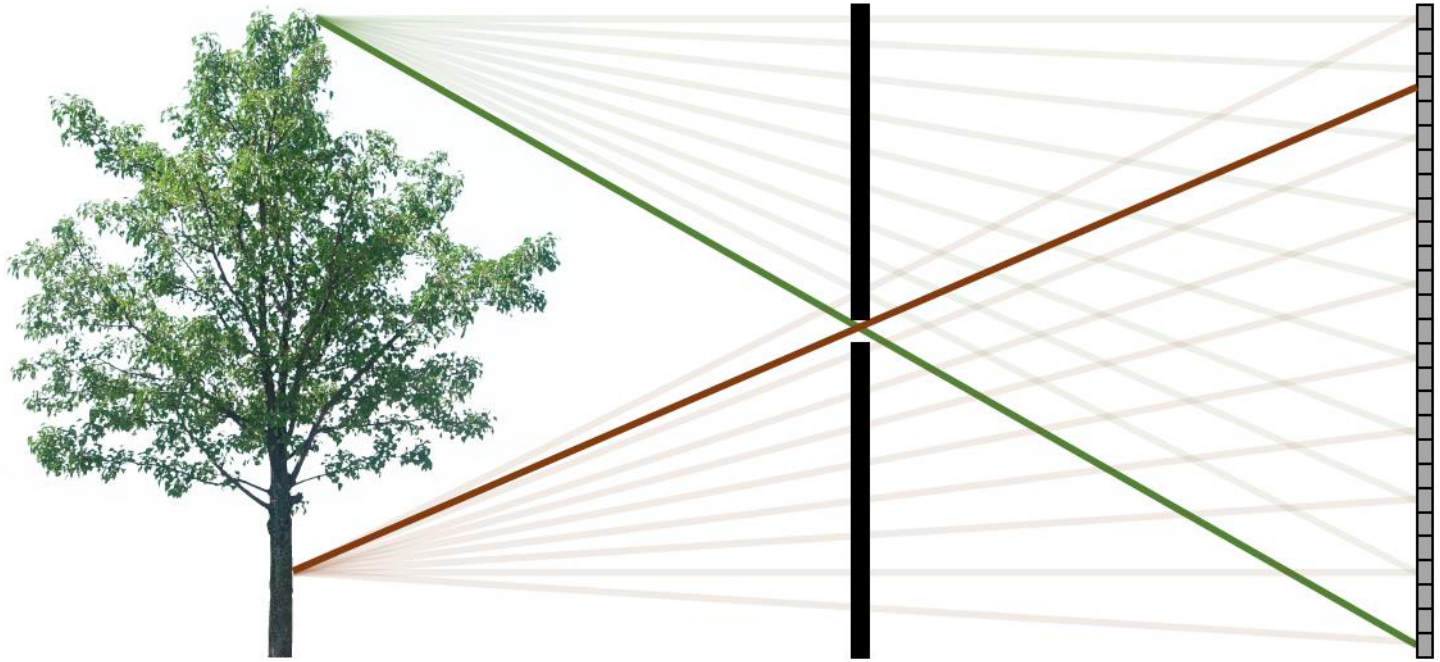
Bare-Sensor Imaging



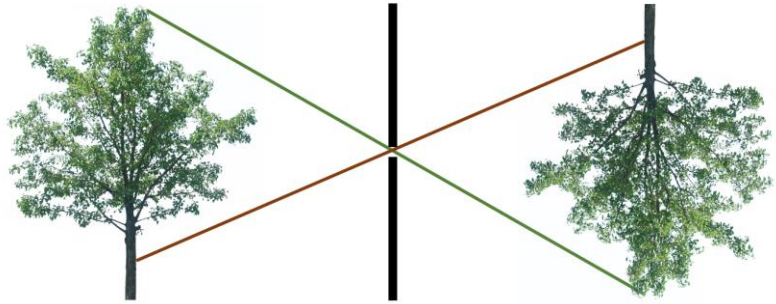
all scene points contribute to all sensor pixels:

Pinhole Camera

camera: barrier with
pinhole (aperture)

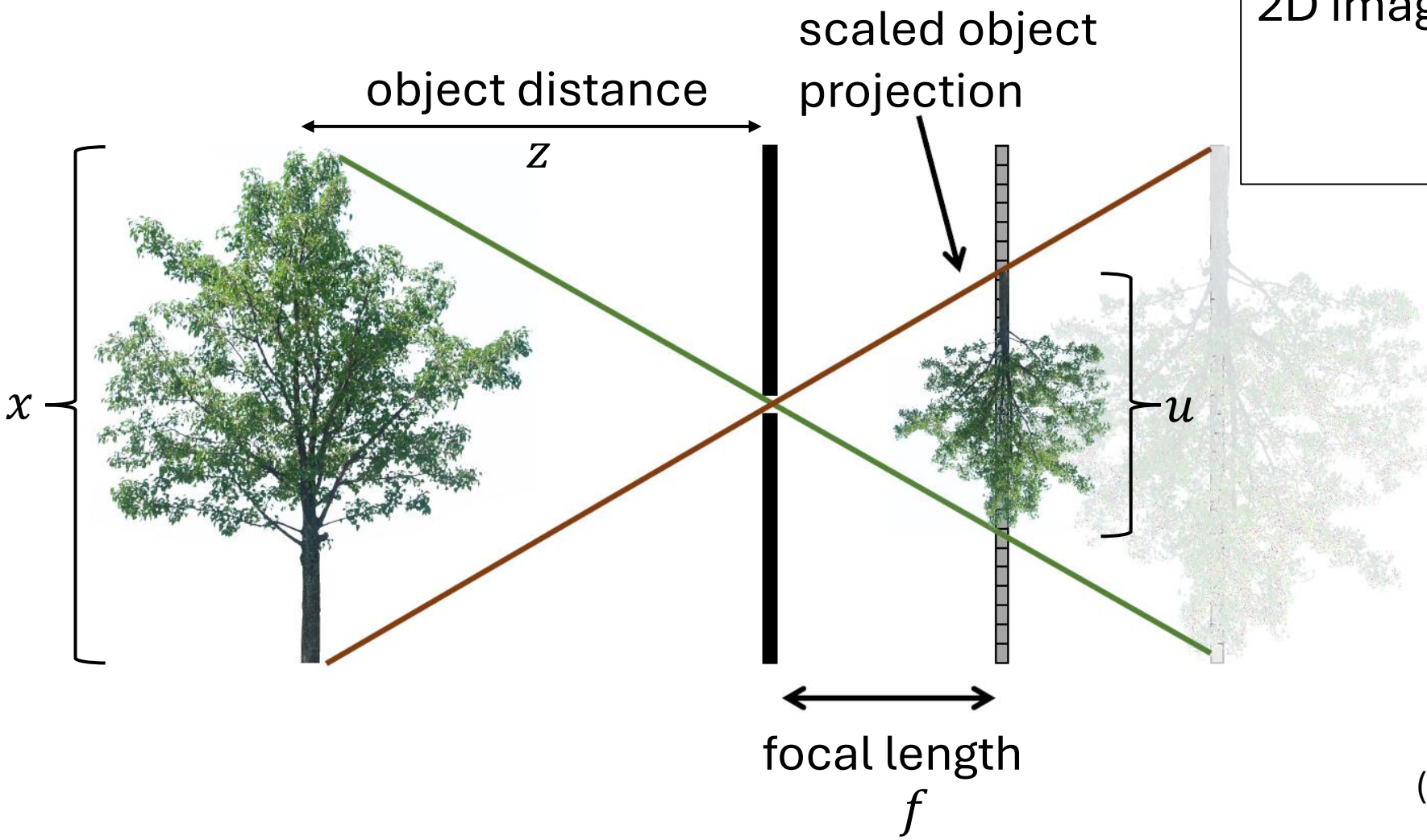


camera obscura



each scene point contributes
to one pixel only

Pinhole Camera Model



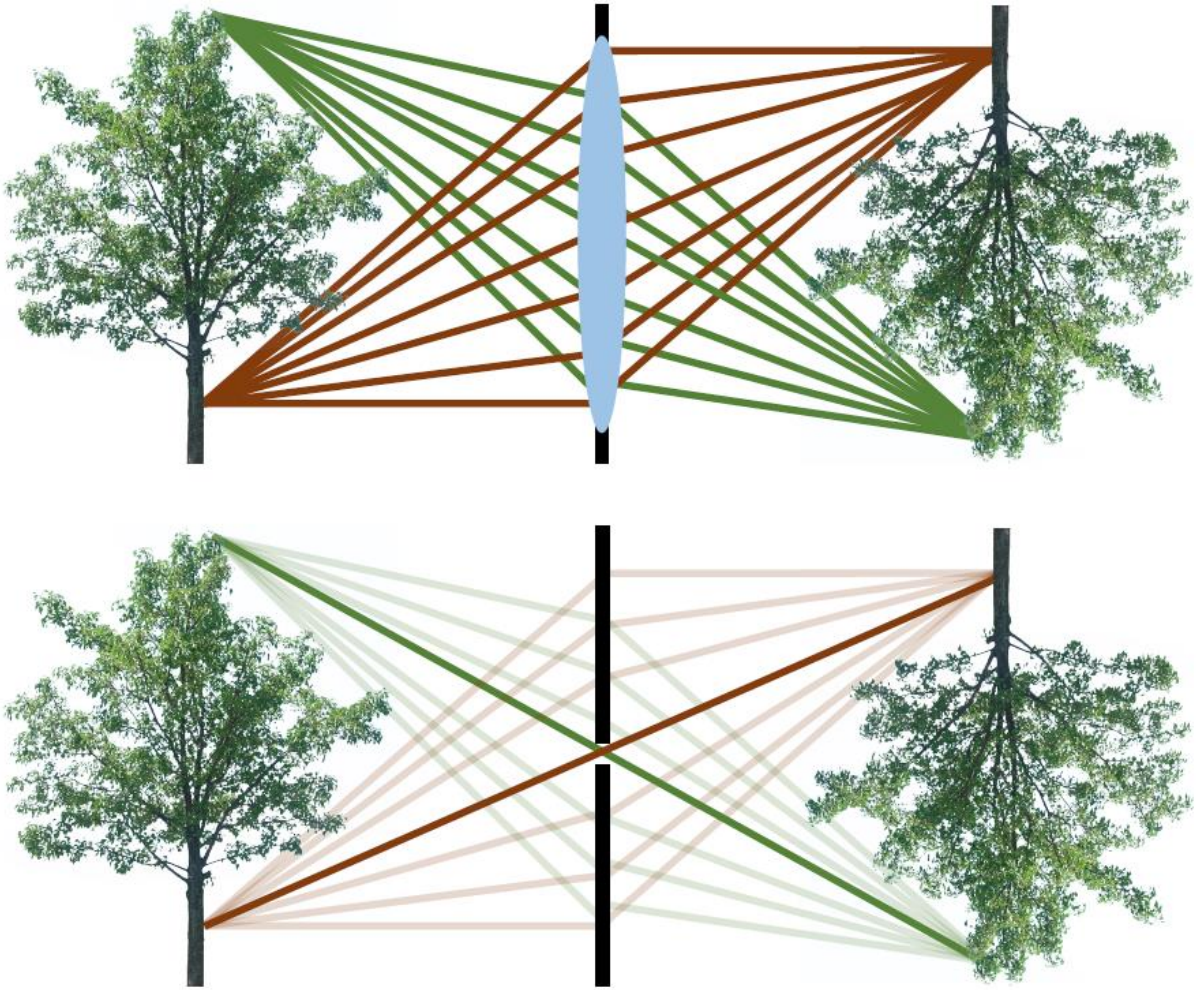
3D world coordinates (x, y, z) to 2D image coordinates (u, v):

$$\begin{pmatrix} u \\ v \end{pmatrix} = -\frac{f}{z}\begin{pmatrix} x \\ y \end{pmatrix}$$

losing depth

(e.g., stereo vision to get it back)

Lens Camera

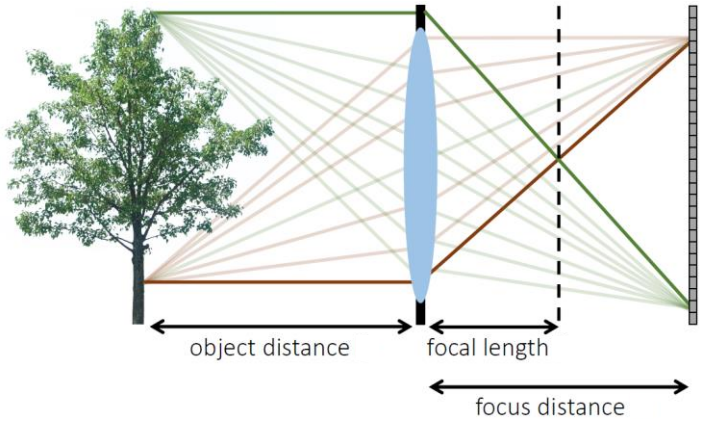


ideal pinhole is infinitesimally small: larger diameter/aperture makes image blurrier (but also allows more light to pass through: higher intensity of image pixels)

→ use optical lens

lens camera can be described with pinhole camera model, if:

- only central rays are considered
- lens camera is in focus (focus distance of lens camera equal to focal length of pinhole camera)



Digital Image Sensors

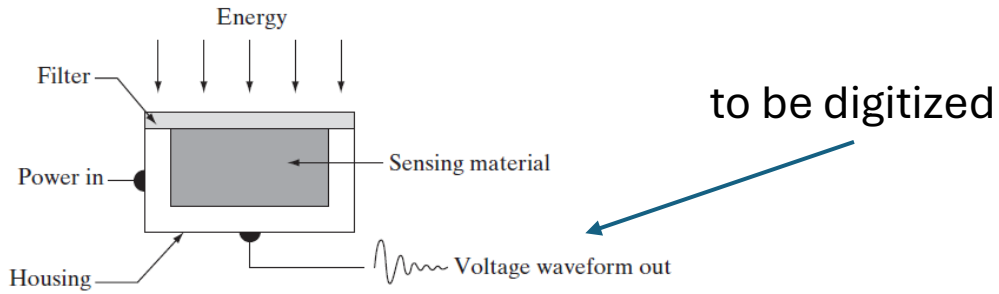
Photons to Electrons

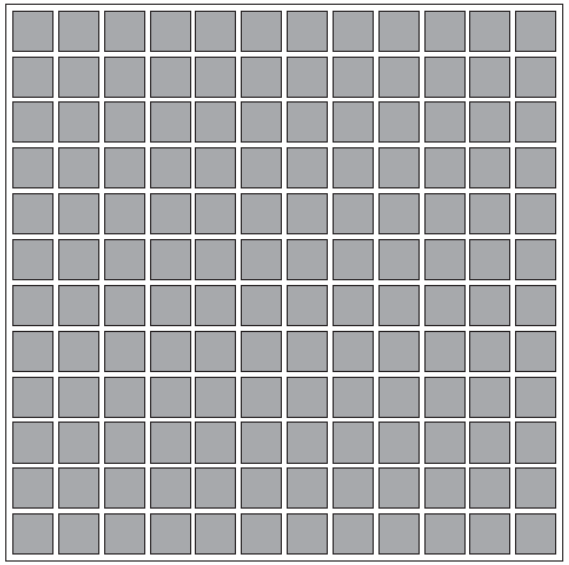
largely replaced chemical and analog imaging

single sensor: photodiode

sensor strips: motion perpendicular to strip for imaging in other direction, used in CT

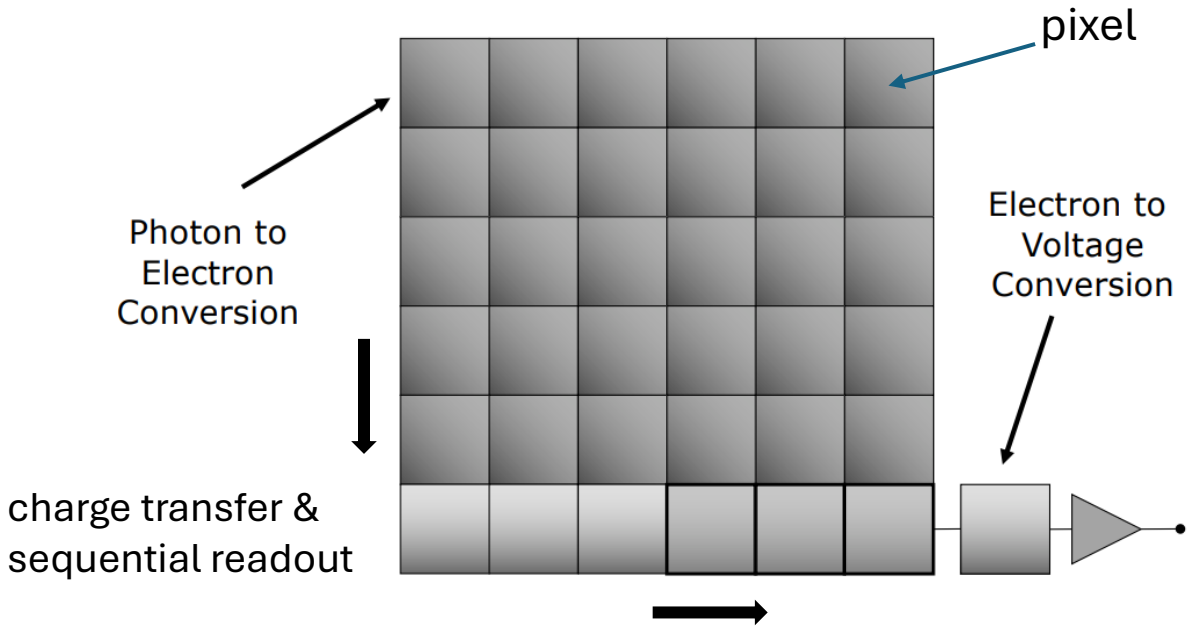
sensor arrays: used in digital cameras (in focal plane, outputs proportional to integral of light received at each sensor)





Most Popular Sensors: CCD and CMOS

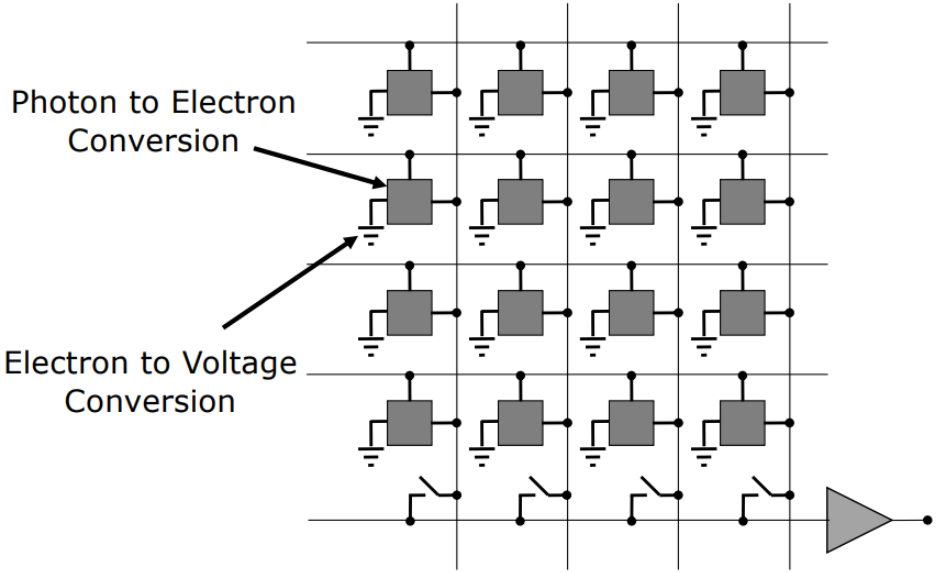
CCD: Charge Coupled Device



larger light-sensitive areas, lower noise level
→ used in high-end video cameras

active pixel sensor

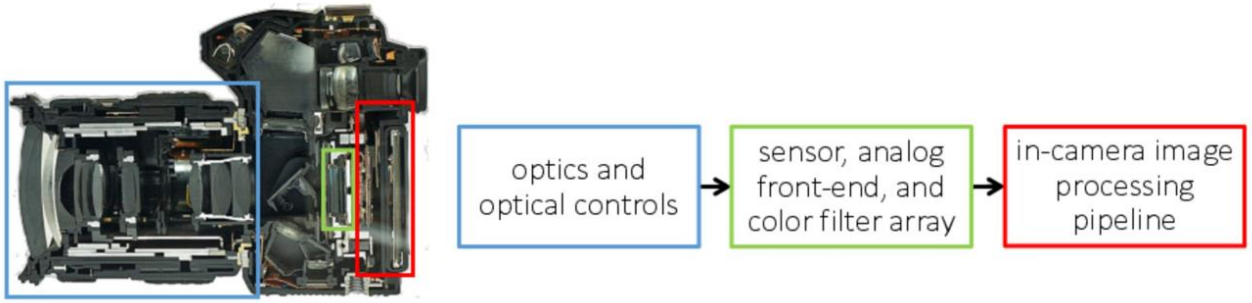
CMOS: Complementary Metal-Oxide Semiconductor



lower power consumption, typically smaller
→ used in most consumer product cameras

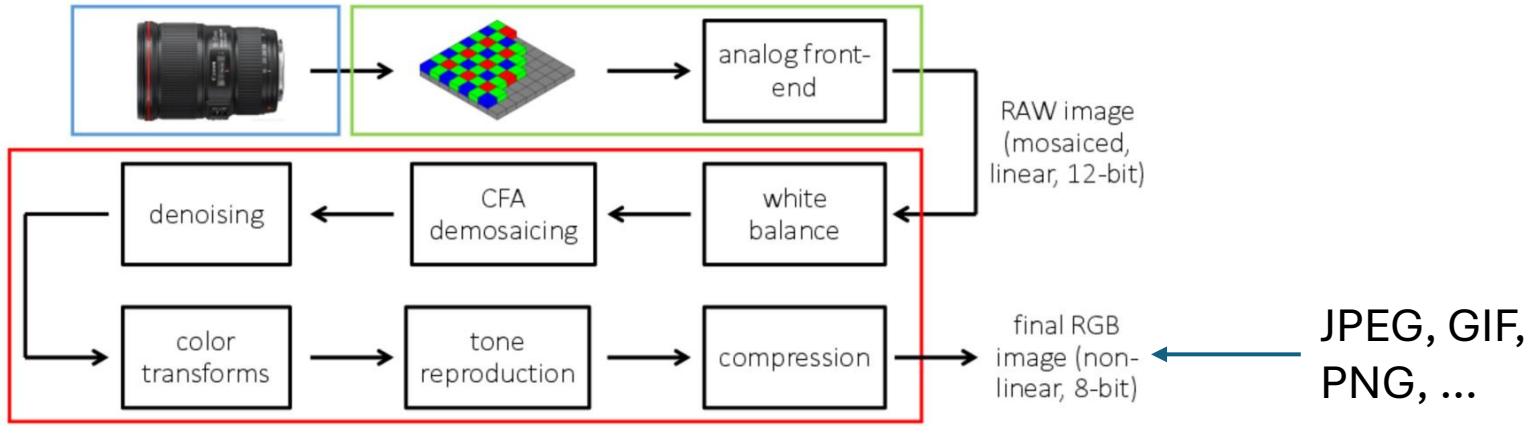
no individual silicon cells, but microlenses on top of each pixel to focus light on sensitive areas

The modern photography pipeline



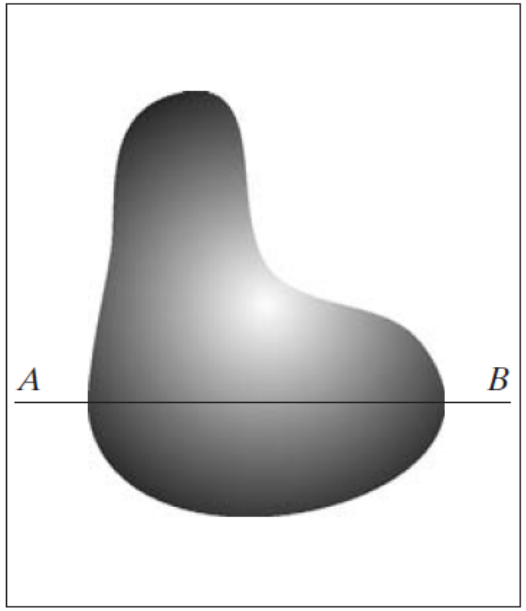
1. camera shutter opens
2. photodiodes: photons to electrons, store in potential well
3. camera shutter closes

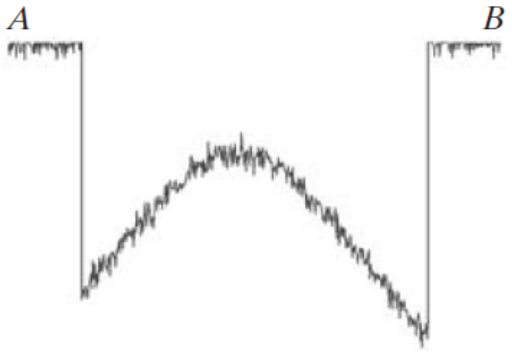
4. readout of potential wells, conversion to analog signals
5. conversion to digital signals (intensity levels)

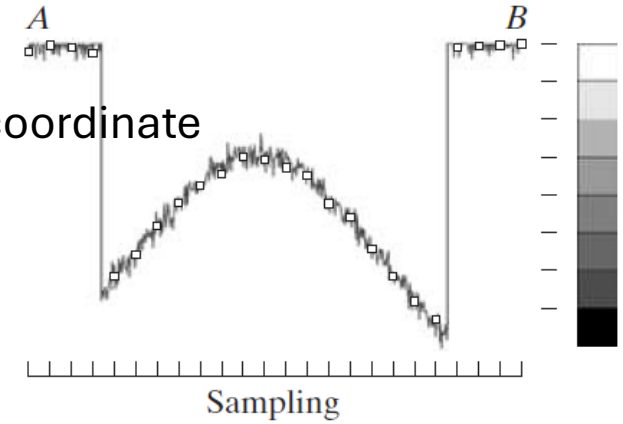


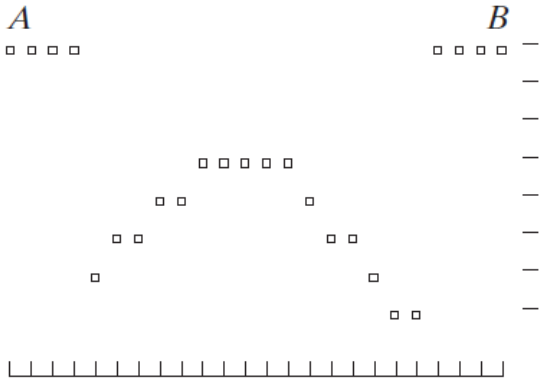
Pixels & Intensity

Digitizing: Sampling & Quantization

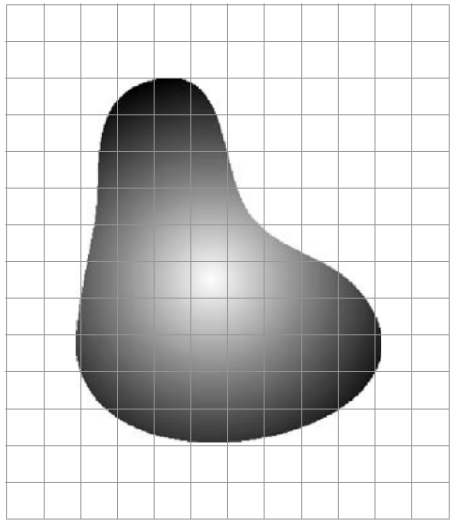


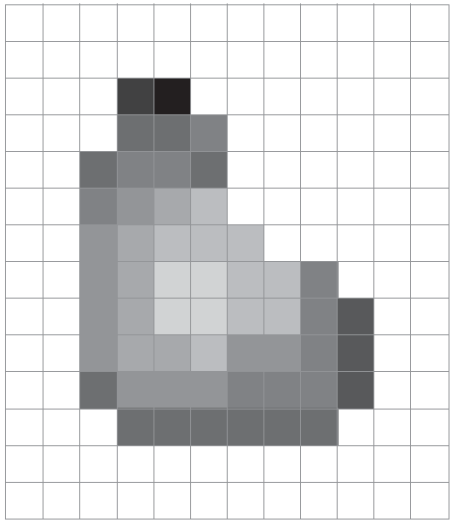






digitizing the coordinate values: pixels





digitizing the amplitude values: discrete intensity levels

Discrete Intensity Levels

image data: 2D grid (matrix) of pixels with different intensity values

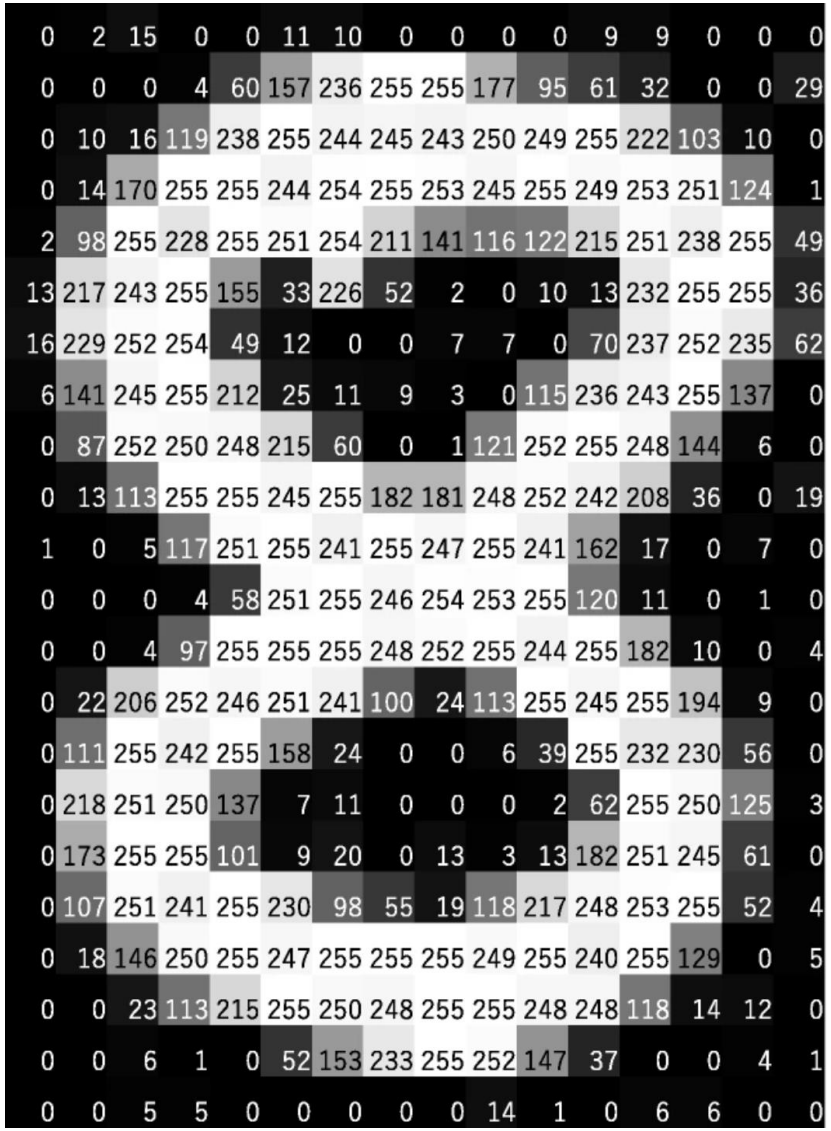
intensity resolution:

common to use one byte (2^8) to express possible intensity values (8-bit image)

→ 0 = black, 255 = white

storage requirement for, e.g., 1024×1024 8-bit image (grayscale): ~1 MB

($\times$ 3 for RGB color images)



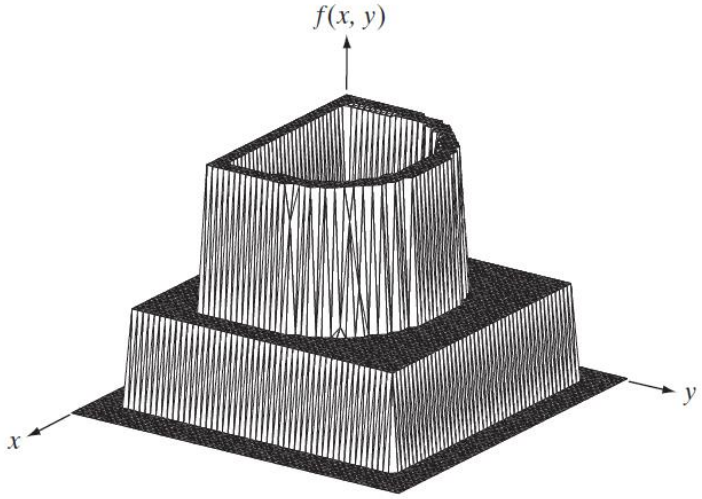
source

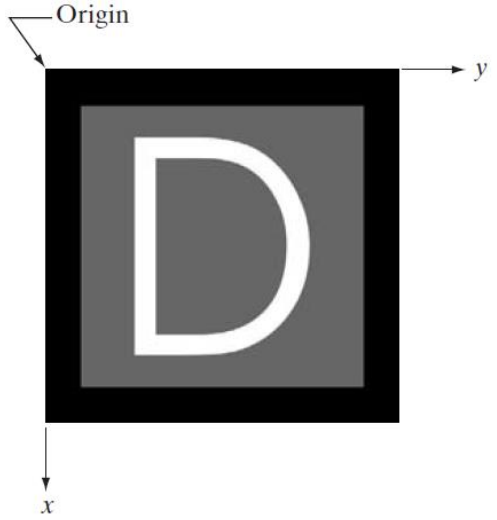
Image as Two-Dimensional Function

image can be seen as function $f(x, y)$: intensity at position (x, y)

digital image: discrete (sampled and quantized) version of this function

$$f(x, y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0, N-1) \\ f(1,0) & f(1,1) & \cdots & f(1, N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1, N-1) \end{bmatrix}$$





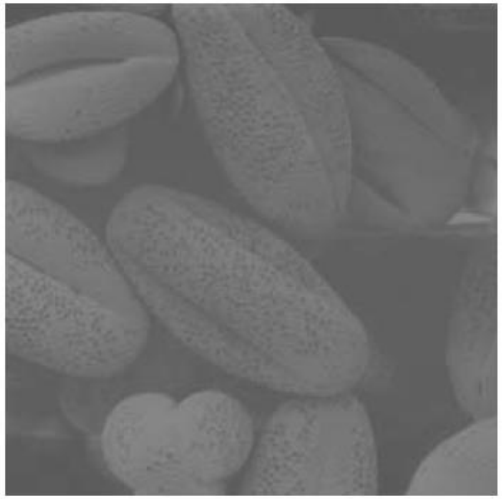
Contrast

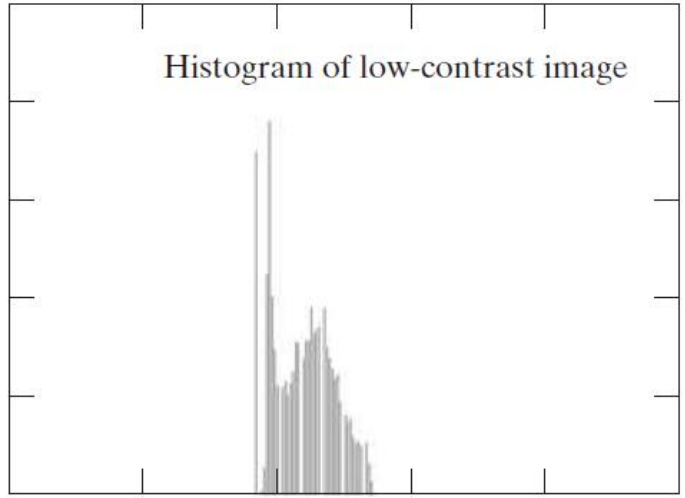
range of intensity levels:

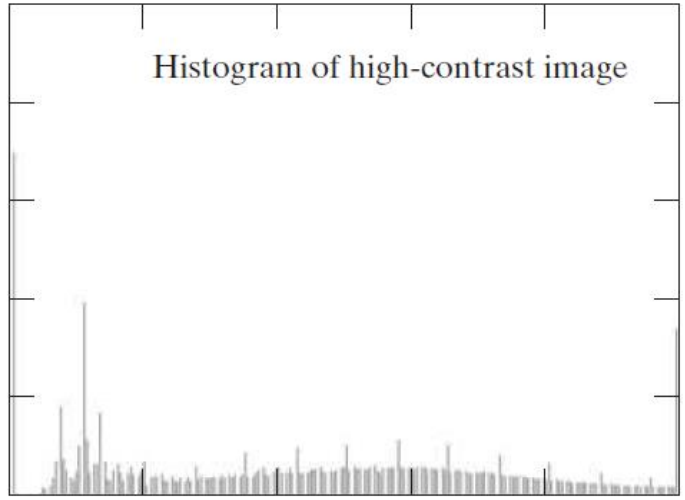
- maximum measurable intensity determined by saturation
- minimum detectable intensity determined by noise

contrast: difference in intensity between highest and lowest intensity levels in an image

intensity histograms:





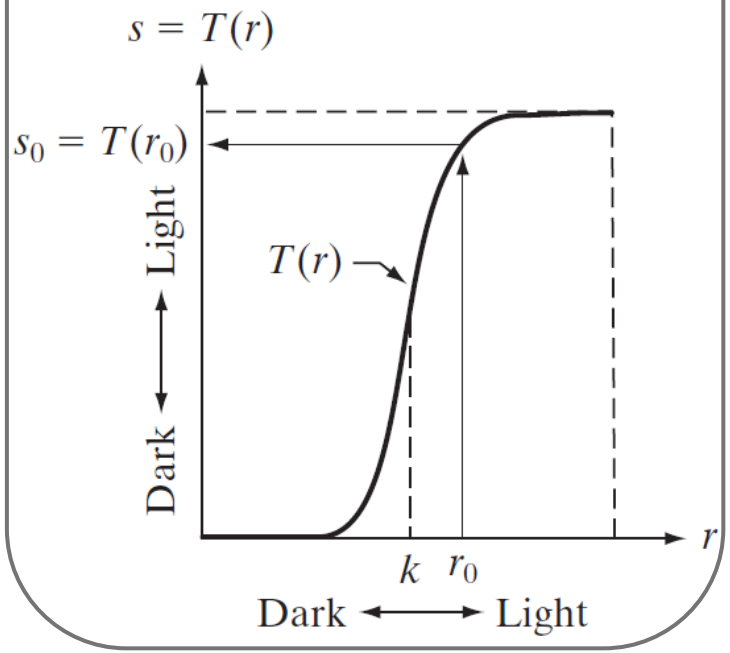


lighten or darken image simply by adding or subtracting a fixed value from the pixel intensities

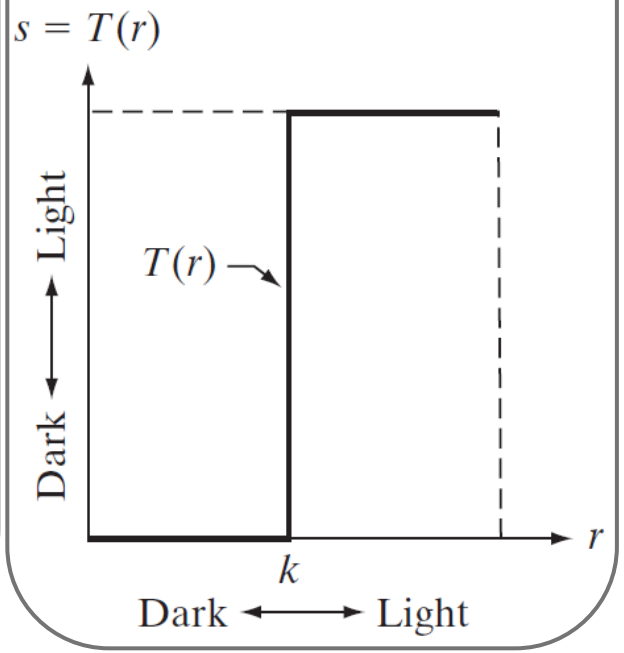
Intensity Transformations

e.g., image enhancement through contrast manipulation

contrast stretching (expanding range of intensity levels):



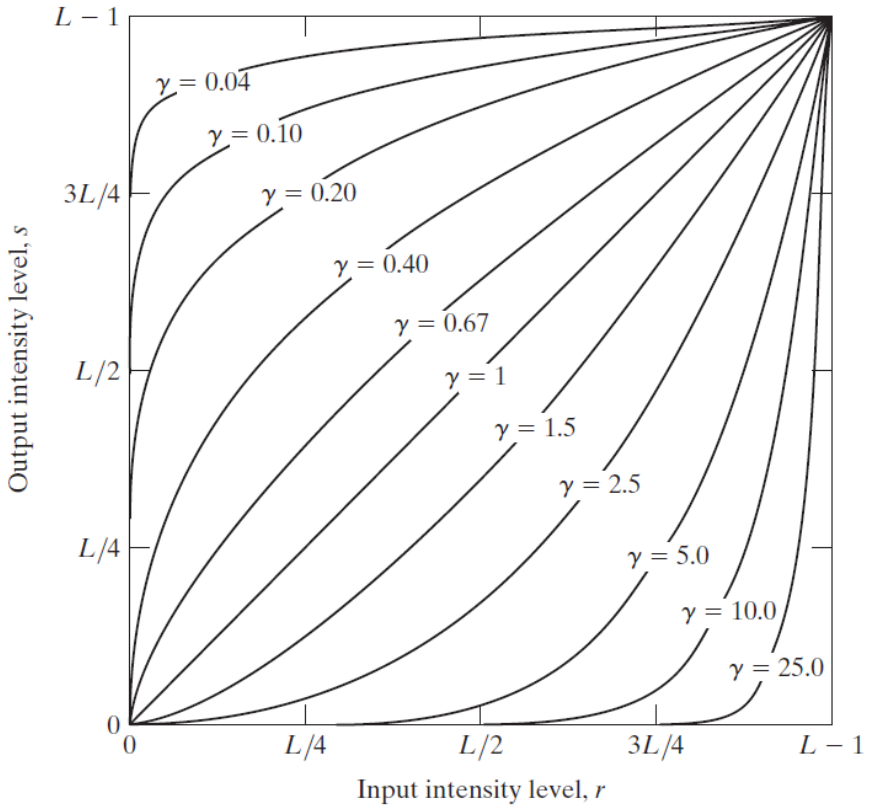
thresholding (differentiation of foreground and background regions):



Power-Law Transformations

simultaneous spreading and compression of intensity levels

e.g., gamma correction:



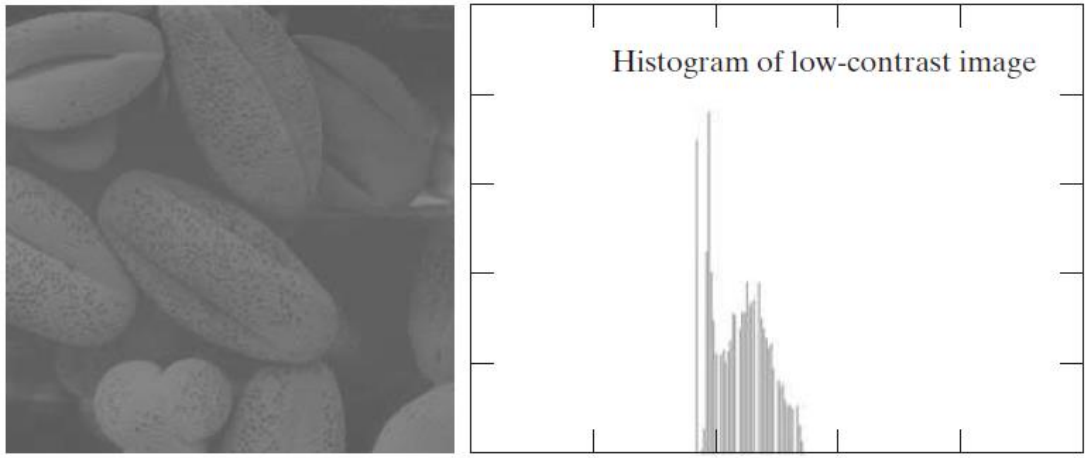
$$s = r^{\gamma}$$

Histogram Equalization

automatic contrast stretching

global transformation function: based on intensity distribution of entire image

local histogram equalization by restricting transformation to pixel neighborhoods



after histogram equalization:

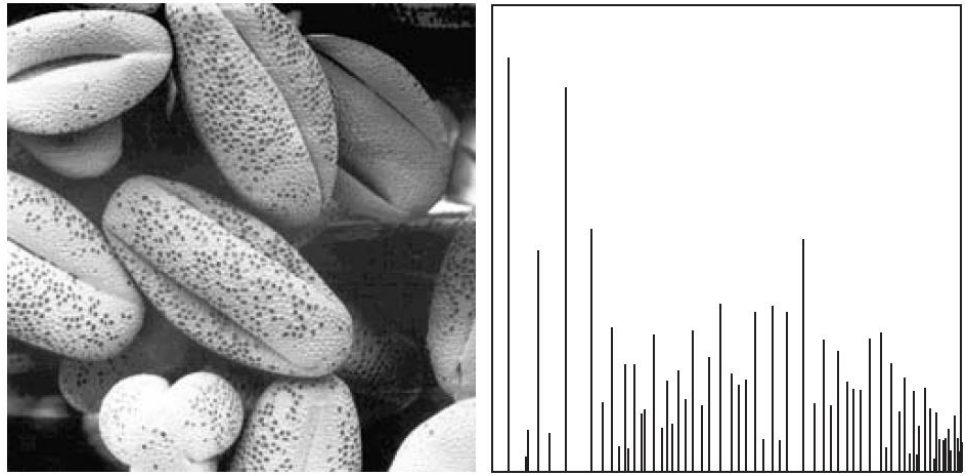


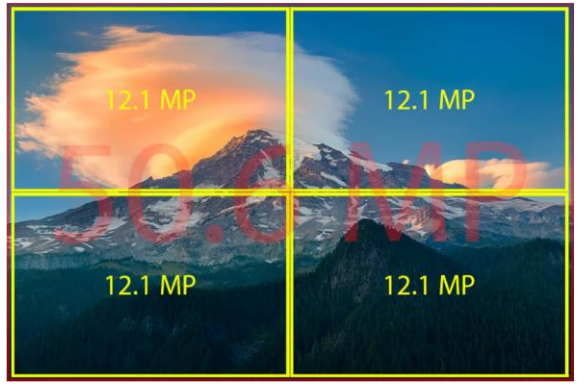
image size: number of pixels

Spatial Resolution

pixels per unit distance: e.g., pixels per inch (ppi)

→ determines print size of digital image with given pixel count

same resolutions



look at images in screen resolution (property of digital screen)

e.g., iPhone 16 Pro Max: 2868×1320 pixels, 6.9 inches diagonal, 460 ppi

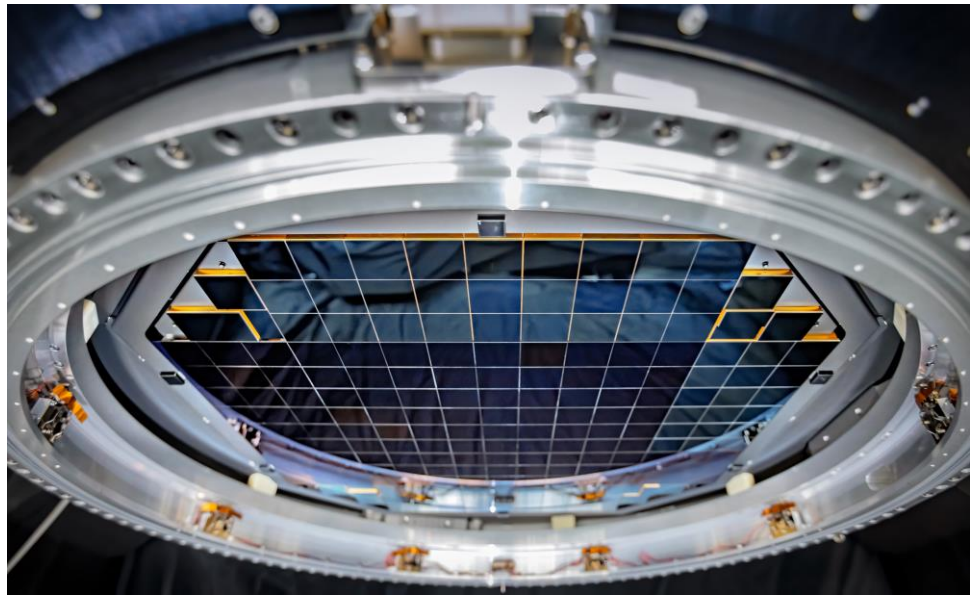
→ either fix number of pixels and calculate figure size, or fix figure size and resample number of pixels (interpolation)

term or printers: dots per inch (dpi) → property of printer

Example from Astronomy

Vera C. Rubin Observatory

first light expected in 2025



largest digital camera ever made: 3.2-gigapixel CCD

Intensity Interpolation

Geometric Transformations

scaling (resizing), rotation, shear, translation, projective warps

approach:

1. relocate individual pixels (without changing its intensity values)
2. calculate new intensity values through interpolation on neighborhood in original image

Example: Resizing

scaling means changing image size in terms of number of pixels

(Do not confuse this with pixel size, which depends only on the used display or printer.)

- shrinking: sub- or downsampling (pixel deletion)
- zooming: upsampling (pixel replication)

original to be upsampled: 

Nearest-neighbor interpolation

Bilinear interpolation

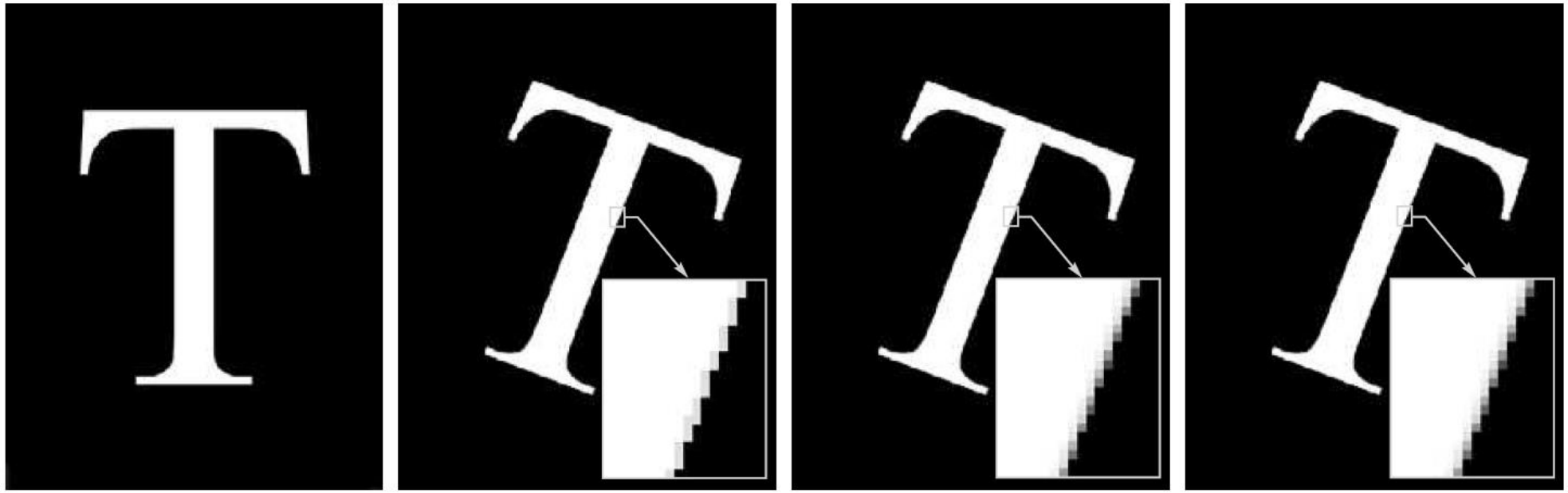
Bicubic interpolation

Example: Rotation

nearest-neighbor interpolation:

bilinear interpolation:

bicubic interpolation:



$v(x,y) = ax + by + cxy + d$
using 4 neighbors

$v(x,y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}x^i y^j$
using 16 neighbors

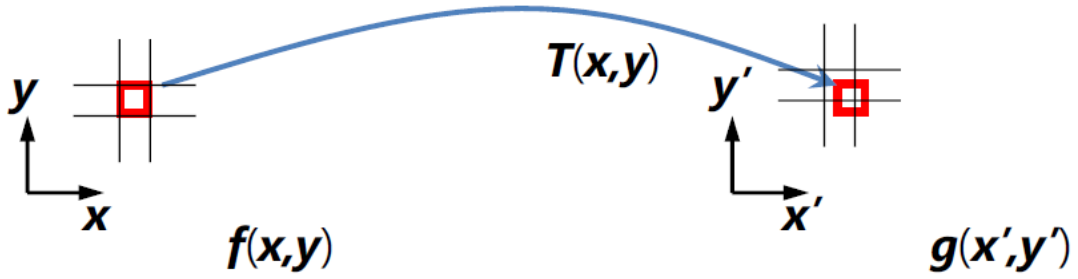
piecewise-polynomial function, aka spline

Mapping

two possibilities:

forward mapping:

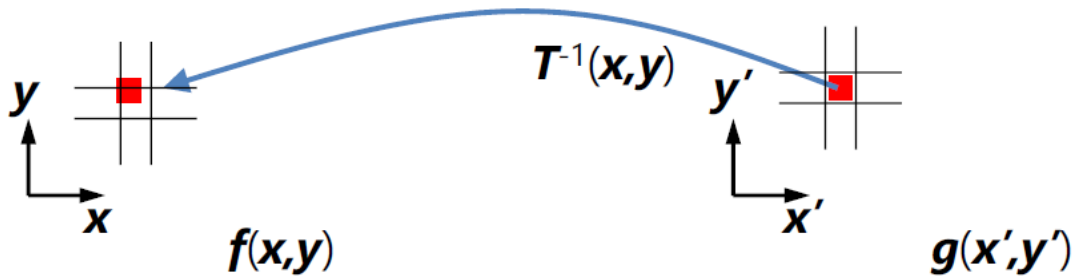
send each source pixel to its target location



each pixel in original image can contribute to more than one pixel in transformed image
→ can result in holes

inverse mapping:

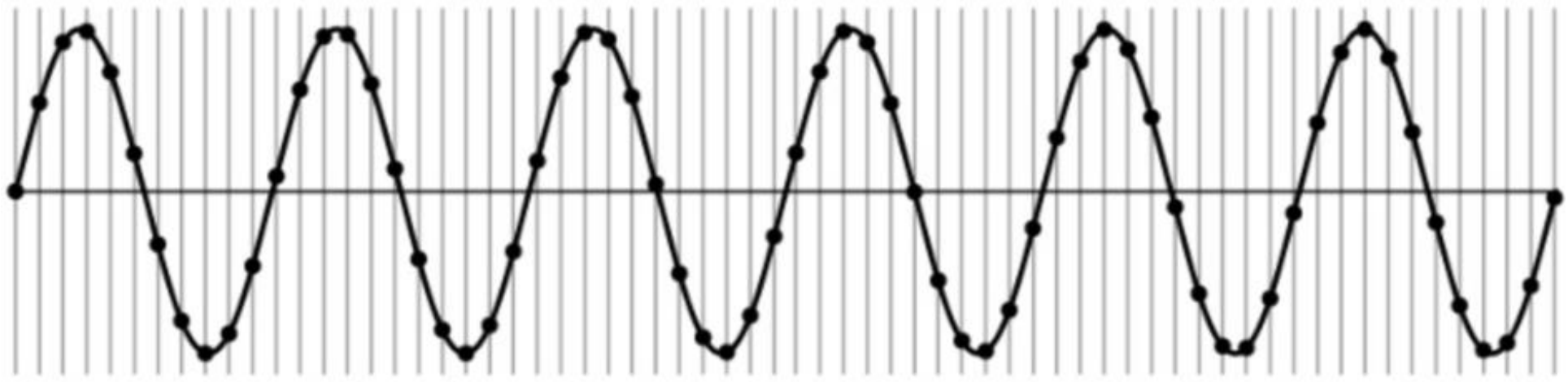
get each target pixel from its source location



each pixel in transformed image can come from more than one pixel in original image
→ resampling from interpolated original image

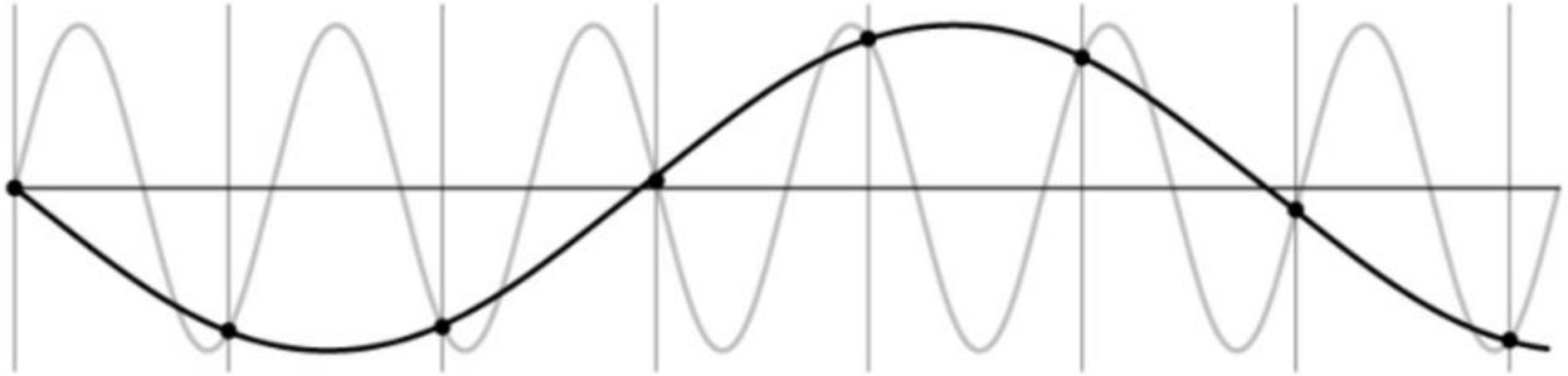
Aliasing

Sampling



remember: digital images sample intensity values at each pixel

Undersampling



aliasing: higher-frequency signal disguised as one of lower frequency

Moiré Effect

aliasing in digital images often manifests itself as ringing patterns

temporal aliasing can be observed as wagon-wheel effect (apparent backward motion) in image sequences (video): frame rate too low with respect to speed of wheel rotation

Anti-Aliasing in Digital Images

no problem if signal is oversampled:

number of pixels (for both width and height) higher than twice the maximum frequency of image (Nyquist frequency)

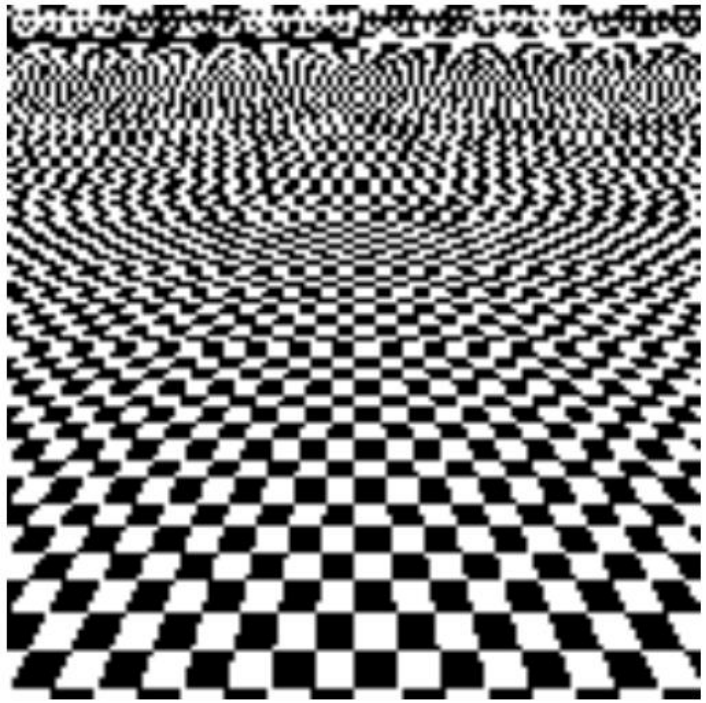
if not oversampled: smooth the signal

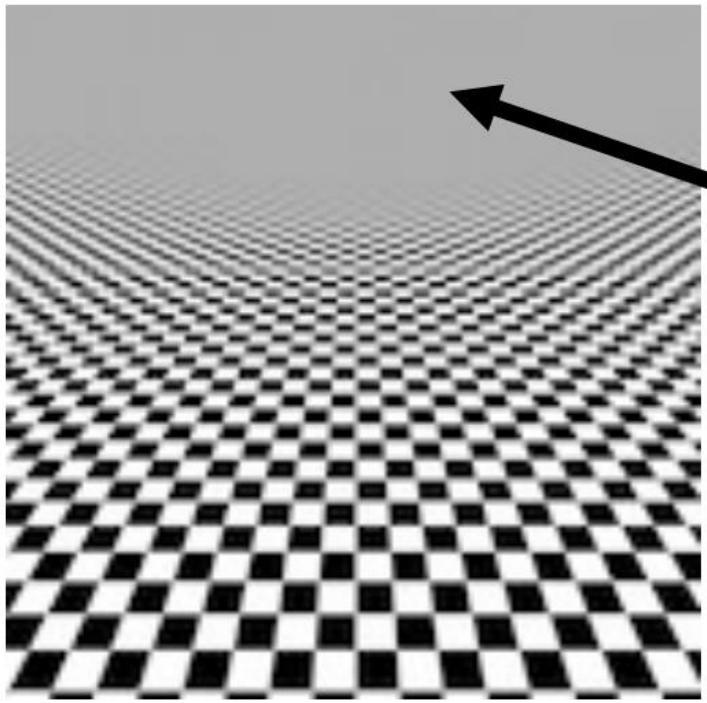
→ removing some details (i.e., high frequencies,) that cause aliasing

optical anti-aliasing filter in digital cameras (on top of image sensor): cut out high frequencies (above Nyquist frequency)

Example: Image Shrinking

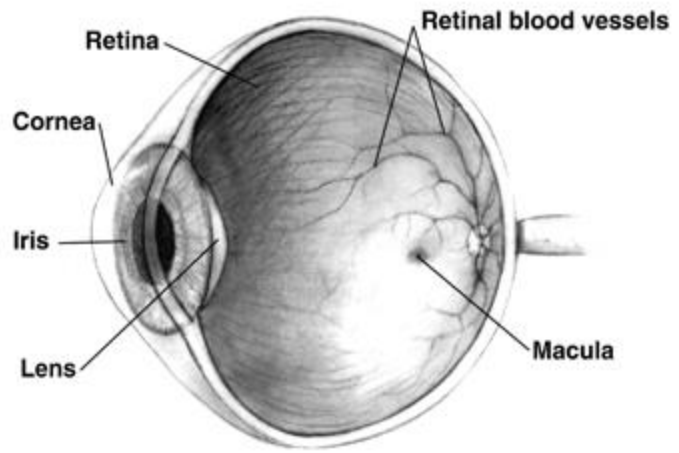
smoothing crucial before subsampling (shrinking) to avoid aliasing: decreases Nyquist frequency of image (to counter less pixels)





Color

Colors in the Human Eye

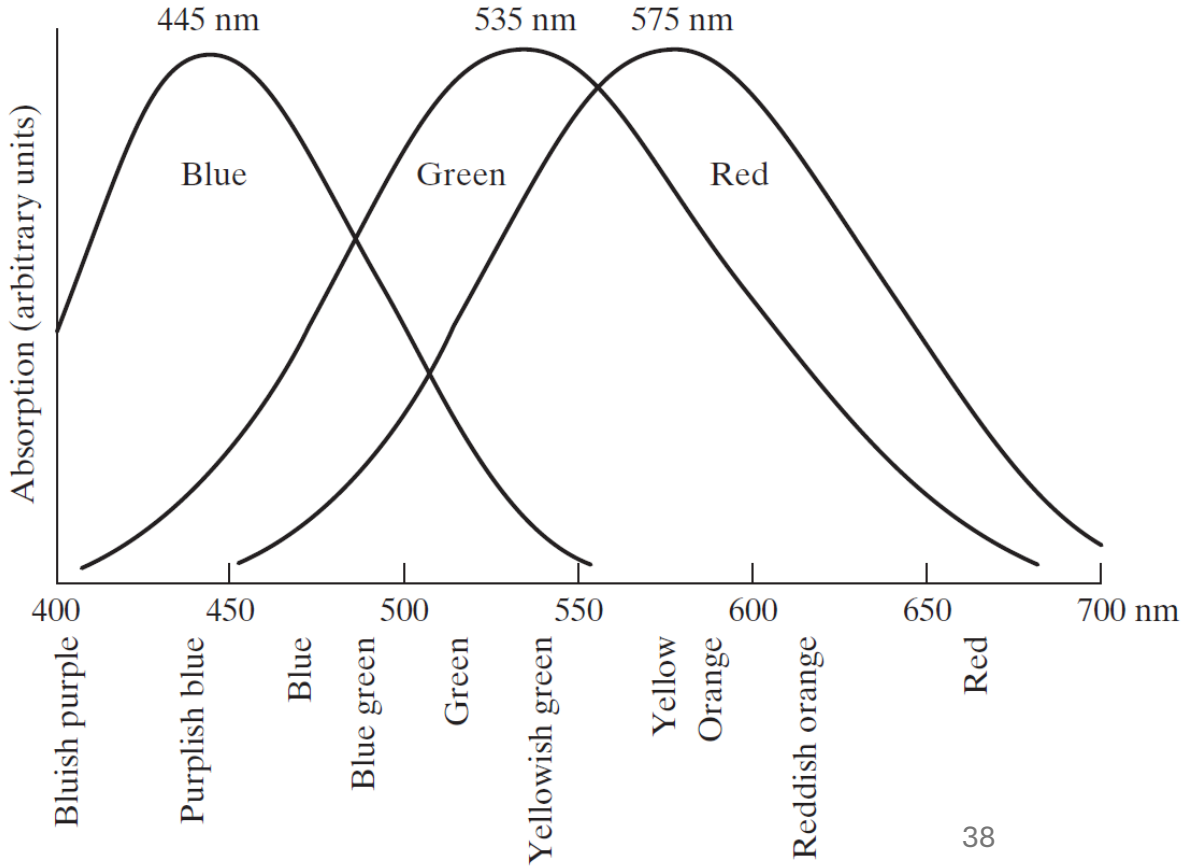


retina: image sensor of human eye

two kinds of light receptors on surface of retina: rods (monochromatic vision) and cones (color-sensitve cells)

three types of cones: red, green, blue

→ combinations of red, green, and blue create perceived colors



Primary Colors

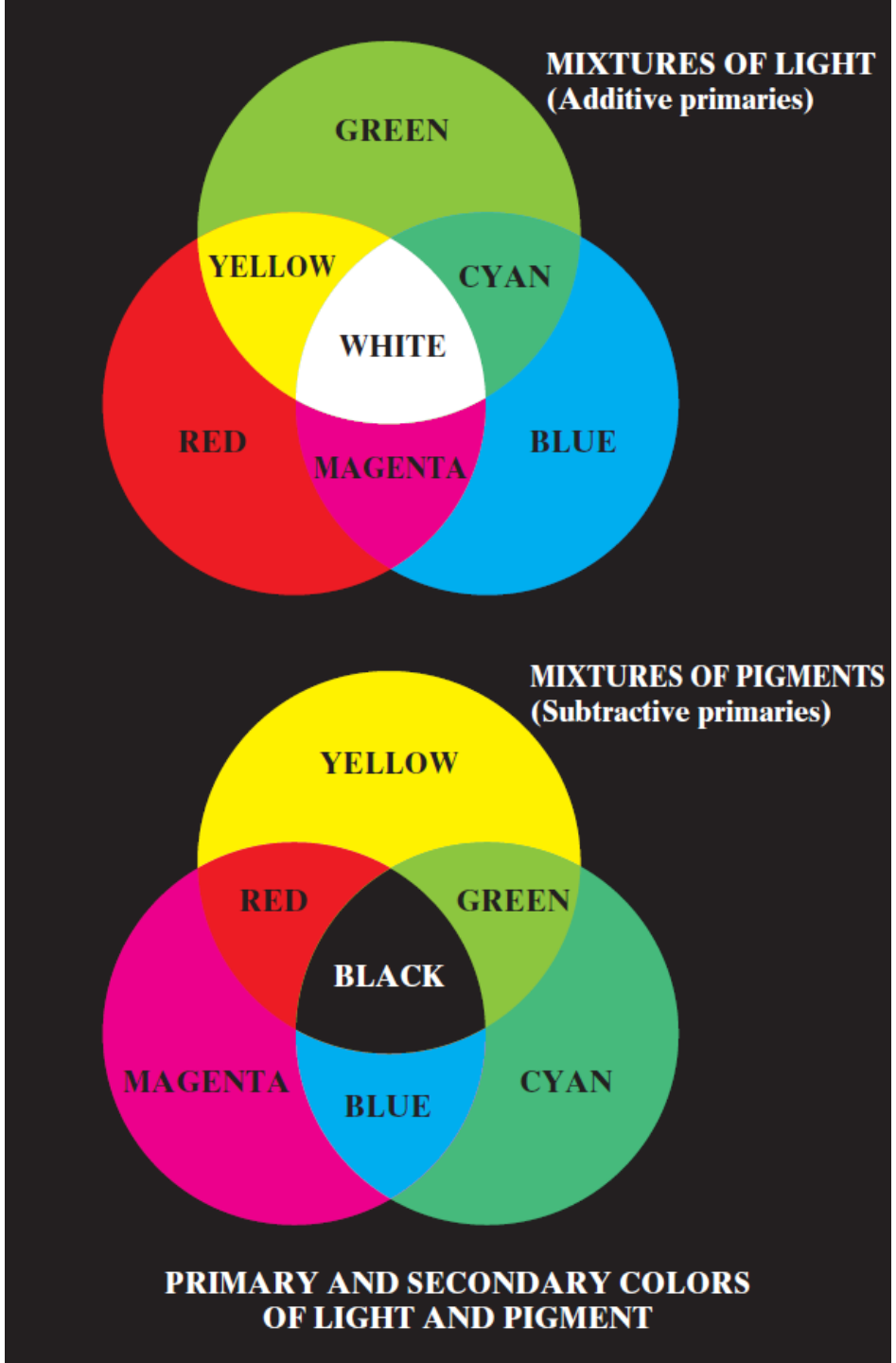
standardization:

use of specific wavelengths for three primary colors red, green, and blue

→ can reproduce (almost) all visible colors by mixing with different intensities

for printing:

primary colors of pigments each absorb one primary color of light

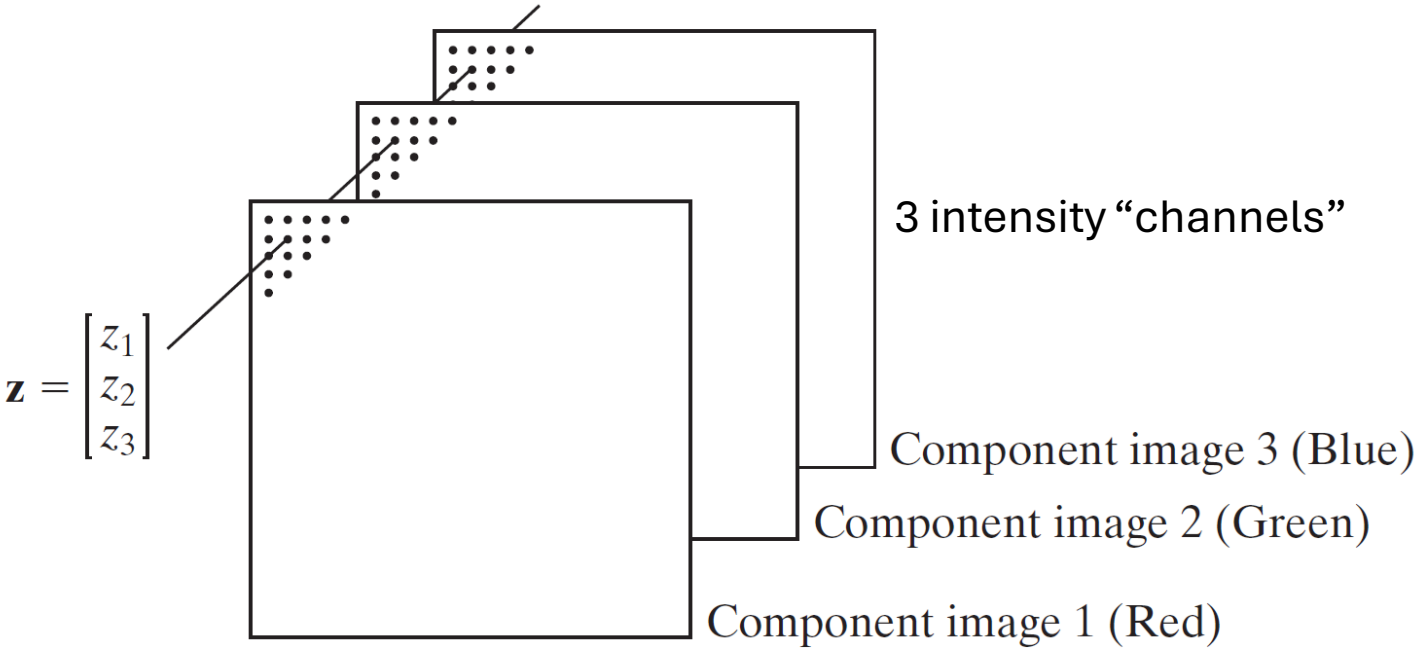


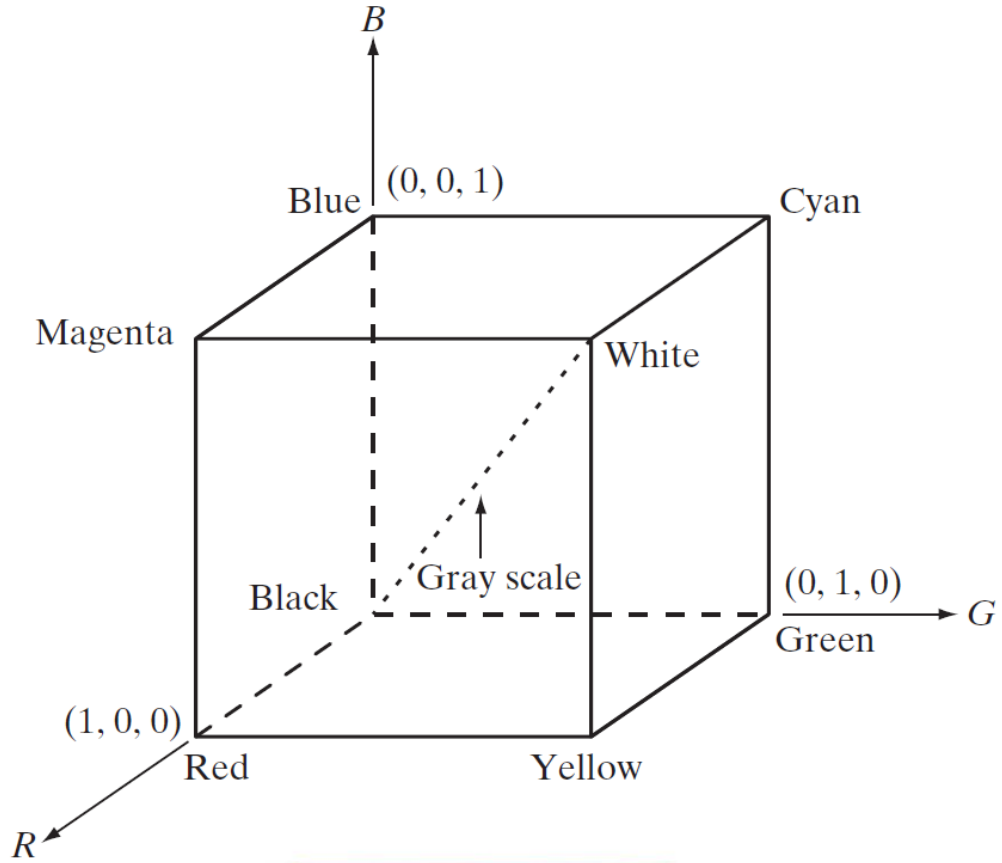
RGB Color Model

color model for monitors and cameras

(other color models: CMYK, HSV, …)

color pixels:



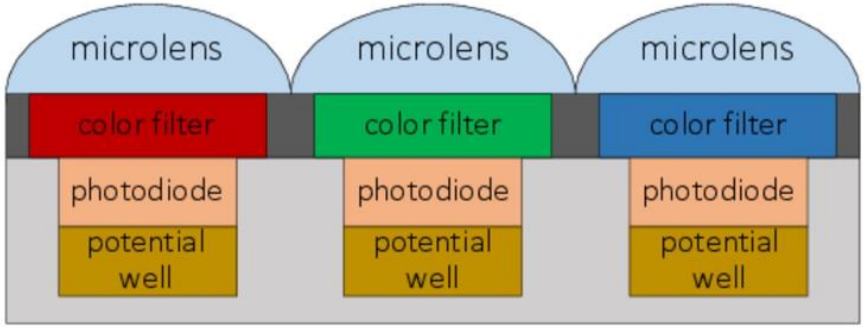


Color Sensing

single image sensor with one of three color filters in front of each pixel

estimation of missing color values for each pixel by interpolation from neighboring pixels

→ full-color image

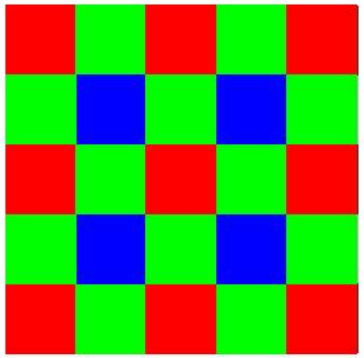


G	R	G	R
B	G	B	G
G	R	G	R
B	G	B	G

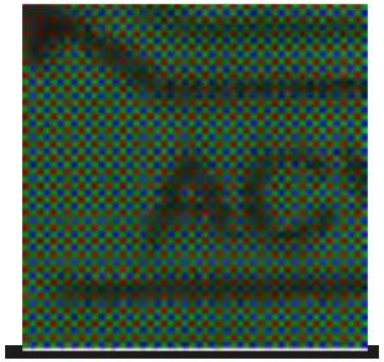
Bayer RGB Pattern

rGb	Rgb	rGb	Rgb
rgB	rGb	rgB	rGb
rGb	Rgb	rGb	Rgb
rgB	rGb	rgB	rGb

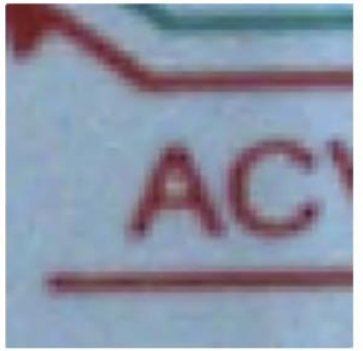
Interpolated Pixels



Bayer Pattern
(Color Filter Mosaic)



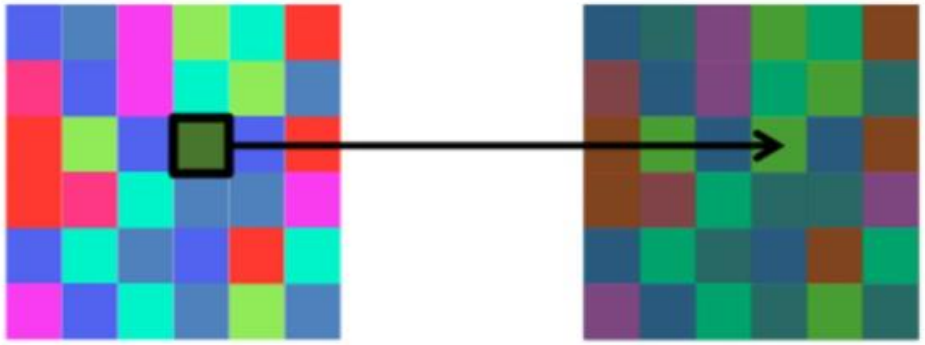
Raw Image



Interpolated Image

Neighborhood Operators

Single-Pixel/Point Operation



point processing

intensity transformations

Neighborhood Operation

“filtering”

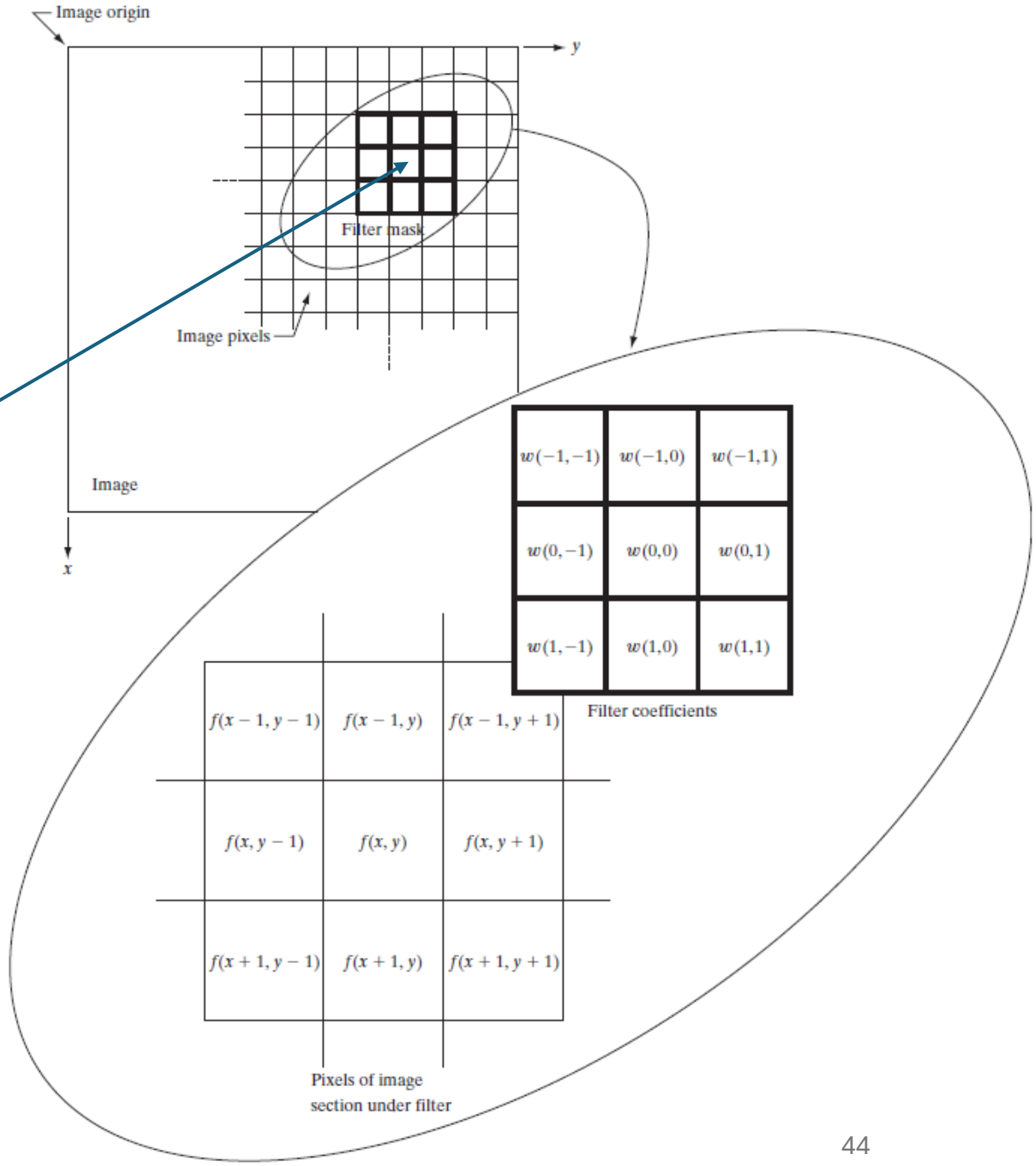
Spatial Filtering

linear shift-invariant operators:

- replace each pixel's intensity by weighted sum of its neighborhood
- weights determined by a kernel (aka mask or filter)
- same kernel **shifted** to all pixels

for example, filter of size 3×3:

$$g(x,y) = \sum_{s=-1}^{1} \sum_{t=-1}^{1} w(s,t) f(x+s, y+t)$$



We set the filter coefficients manually here. Later, we will use a CNN for this.

Example: Box Filter

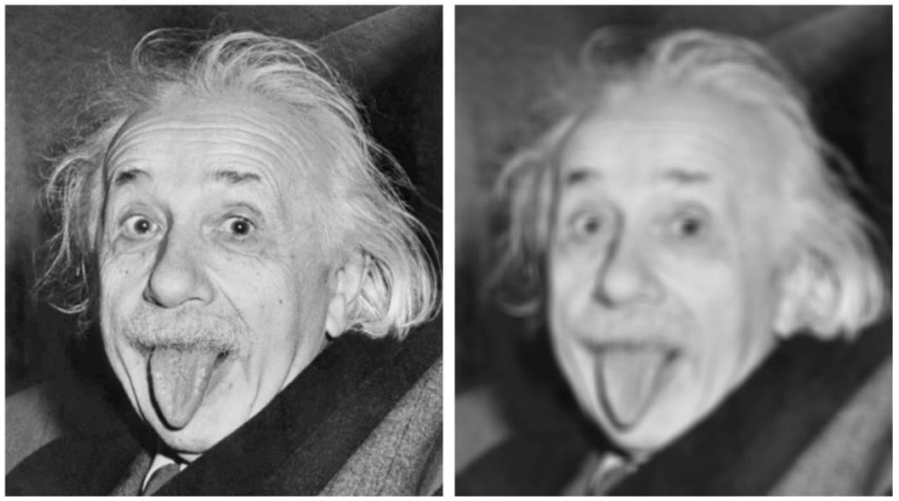
$$w = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$m \times m: w = \frac{1}{m^2}\begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix}$$

the larger the kernel, the heavier the smoothing effect

results in local (moving) averages

→ blurring (smoothing) effect:



- without normalization resulting image would be saturated
- the larger the filter size, the heavier the blurring effect
- filters of even size possible as well, but odd-sized ones easier to deal with

Definition: Correlation vs Convolution

using our example with filter of size 3×3

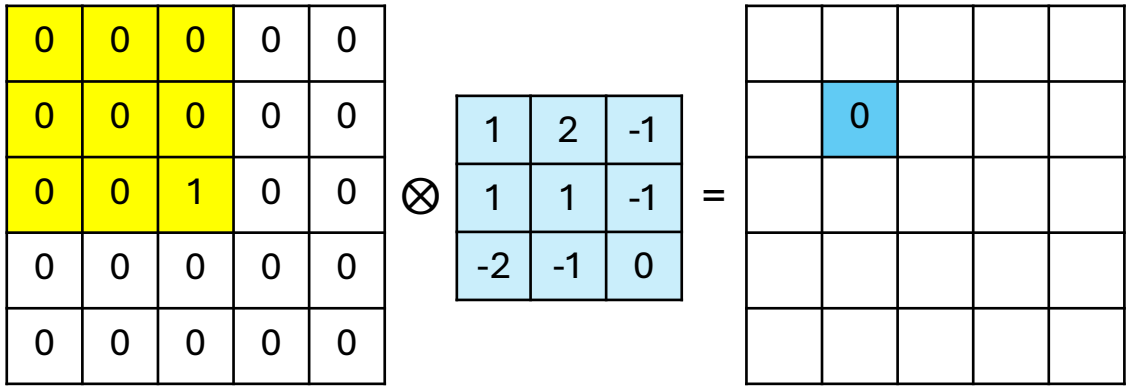
correlation: $g(x,y) = \sum_{s=-1}^{1} \sum_{t=-1}^{1} w(s,t) f(x+s, y+t)$

convolution: $g(x,y) = \sum_{s=-1}^{1} \sum_{t=-1}^{1} w(s,t) f(x-s, y-t)$

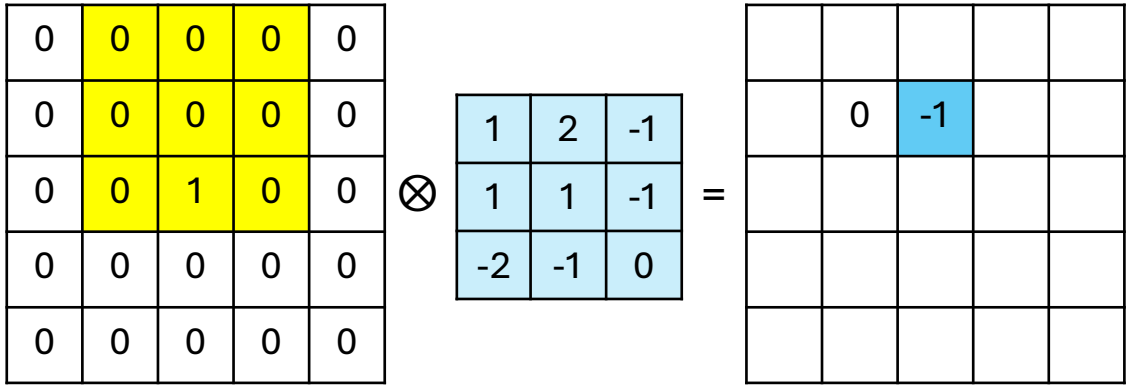
corresponds to flip of filter

- convolution of a kernel with a discrete unit impulse yields copy of the kernel at location of the impulse (and impulse filter yields image copy)
- corresponding correlation instead yields rotated version of the kernel (or rotated image)

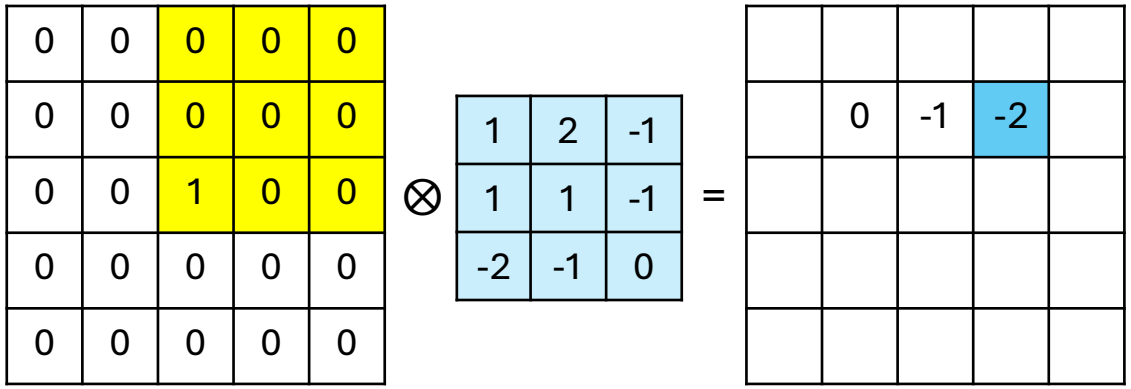
step 1:



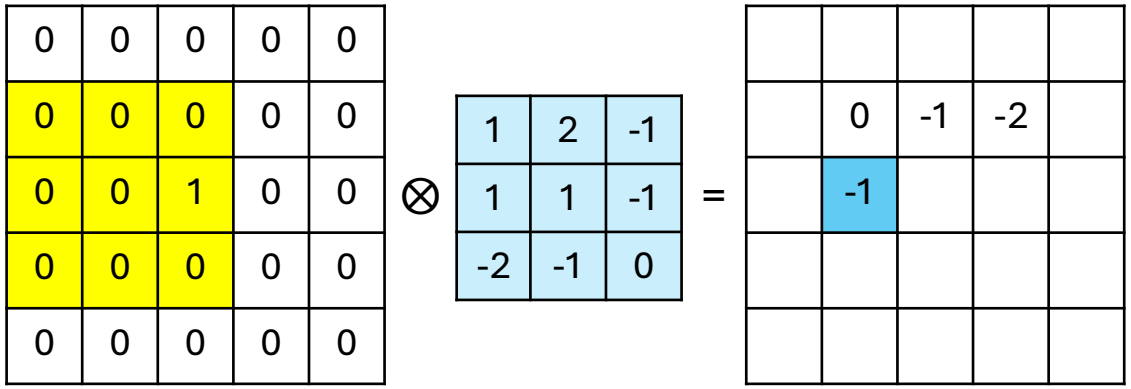
step 2:



step 3:

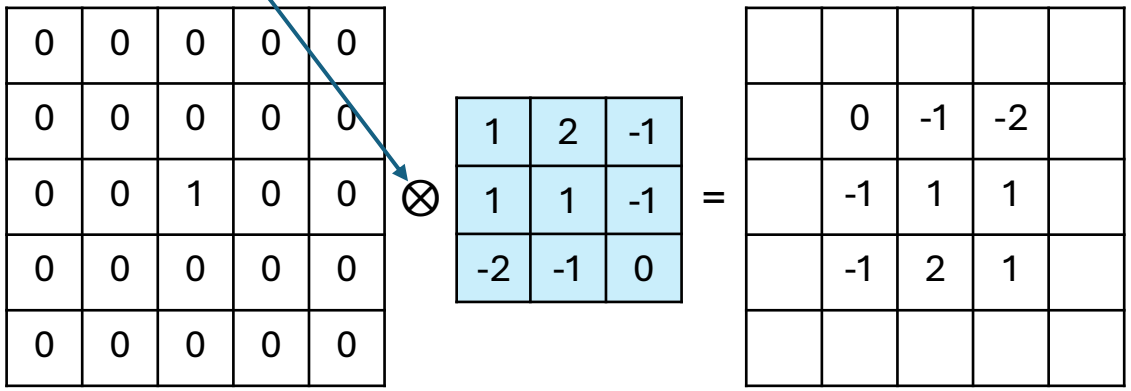


step 4:

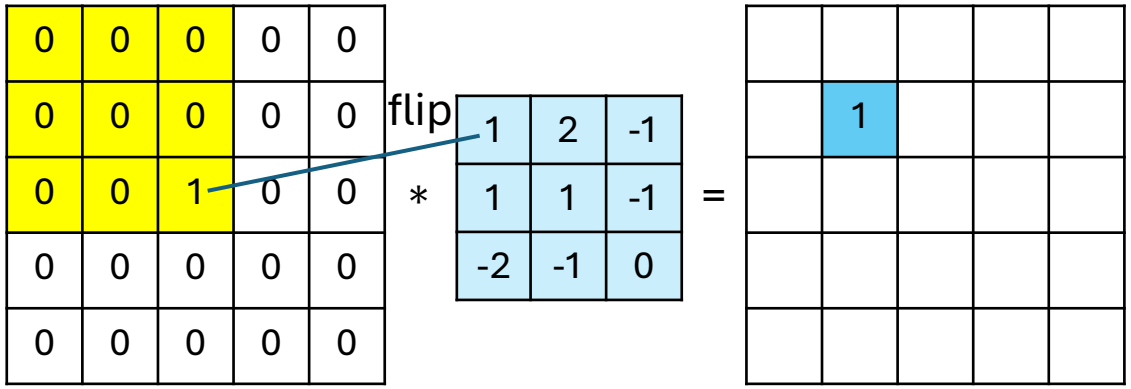


correlation

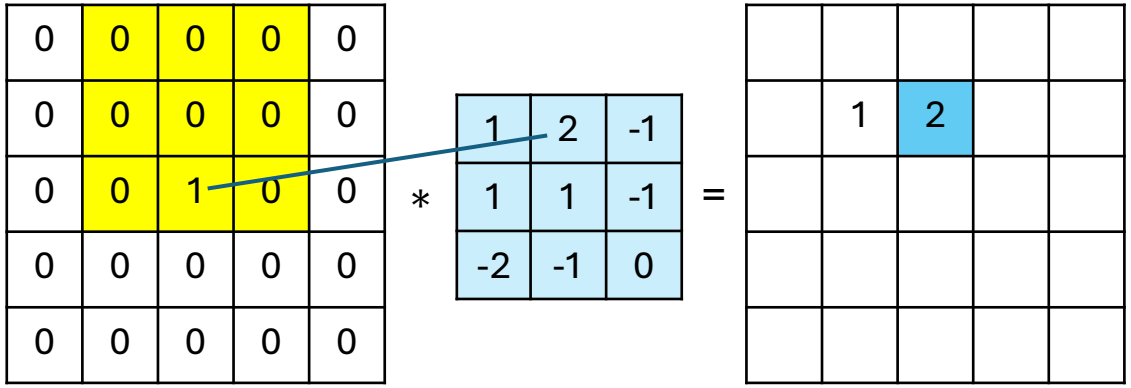
result:



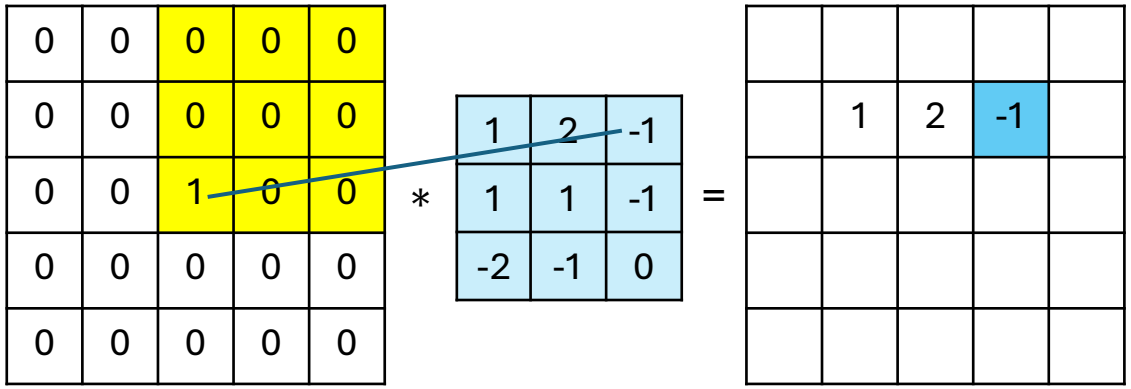
step 1:



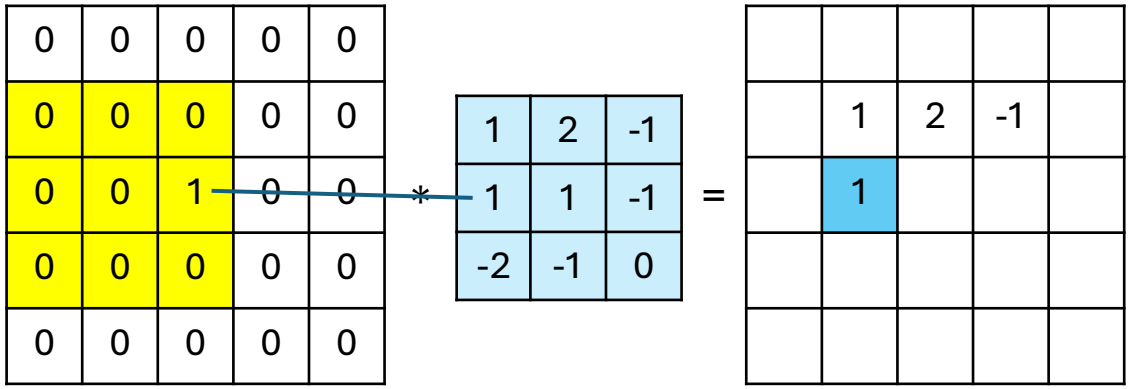
step 2:



step 3:

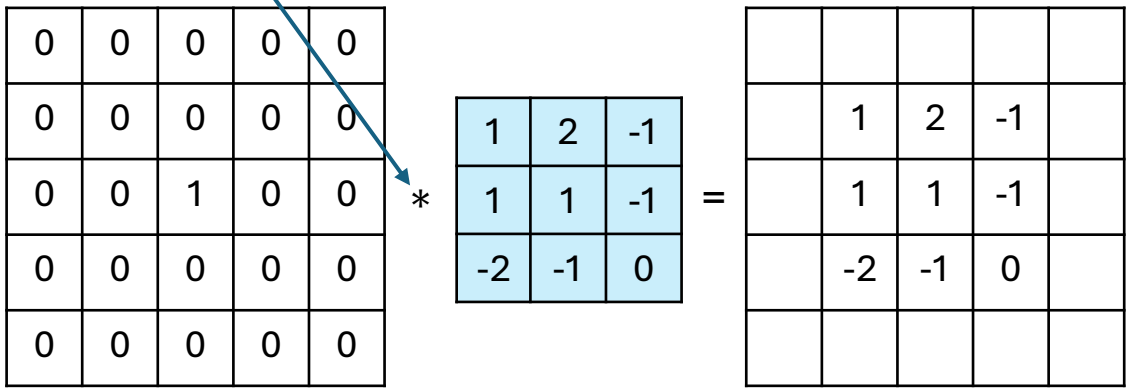


step 4:



convolution

result:



Padding

On last slides, we did not let the filter extend beyond image borders.

→ boundary effects: resulting image smaller than original

zero-padding to avoid this:

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

∗

1	2	-1
1	1	-1
-2	-1	0

=

0	0	0	0	0
0	1	2	-1	0
0	1	1	-1	0
0	-2	-1	0	0
0	0	0	0	0

zero-padding darkens the image edges

other padding options:

- set to constant border value
- repeat edge values
- …

can also be more zero rows/columns (depending on image and filter size)

Other Linear Smoothing Filters

when constructing filters (not only smoothing filters, but in general), look for separable ones:

allow to compute 2D convolution as two subsequent 1D convolutions (horizontal and vertical)

for image of size $M \times M$ and kernel of size $N \times N$:

- $M^2 \cdot N^2$ operations for non-separable filter
- $M^2 \cdot 2 \cdot N$ operations for separable filter

→ significant speedup possible

sampled from 2D Gaussian function (larger σ requires larger kernel)

bilinear filter (tent):

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Gaussian filter:

$$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

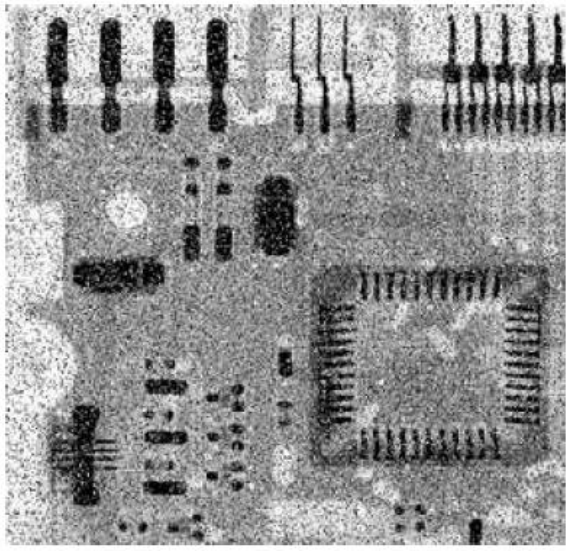
$$\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \qquad \frac{1}{16}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

corresponding horizontal 1D kernels

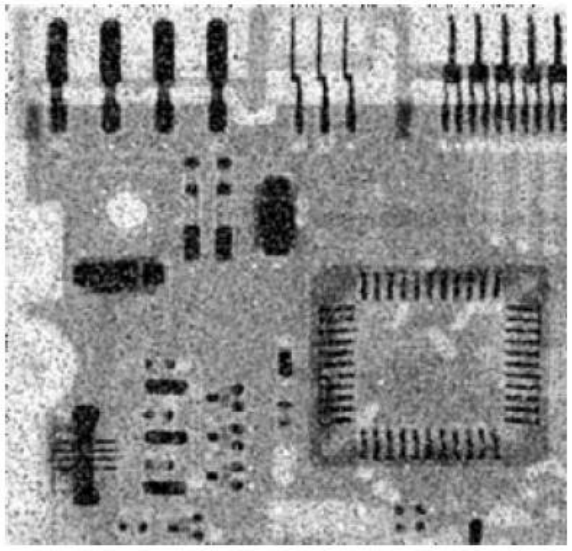
Median Filter

- same kind of neighborhood-shifting over image
- but instead of weighted average, use median of intensity values in pixel neighborhood (no filter coefficients) → non-linear filter
- very effective to reduce outliers, e.g., shot or salt-and-pepper noise

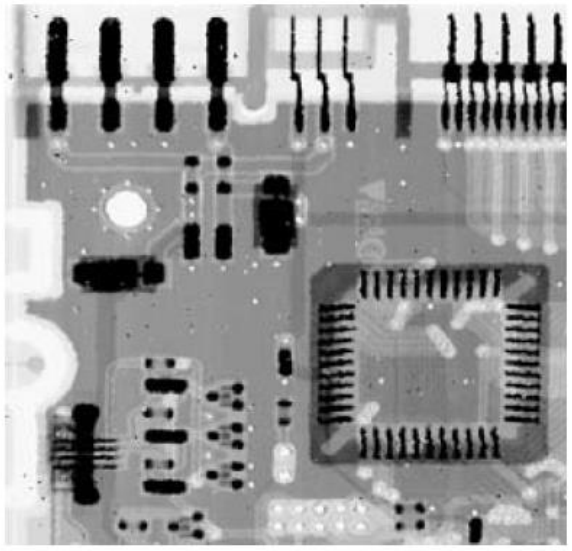
original:



average-filtered (3×3):



median-filtered (3×3):



playground

Sharpening Filters

goal of smoothing filters:

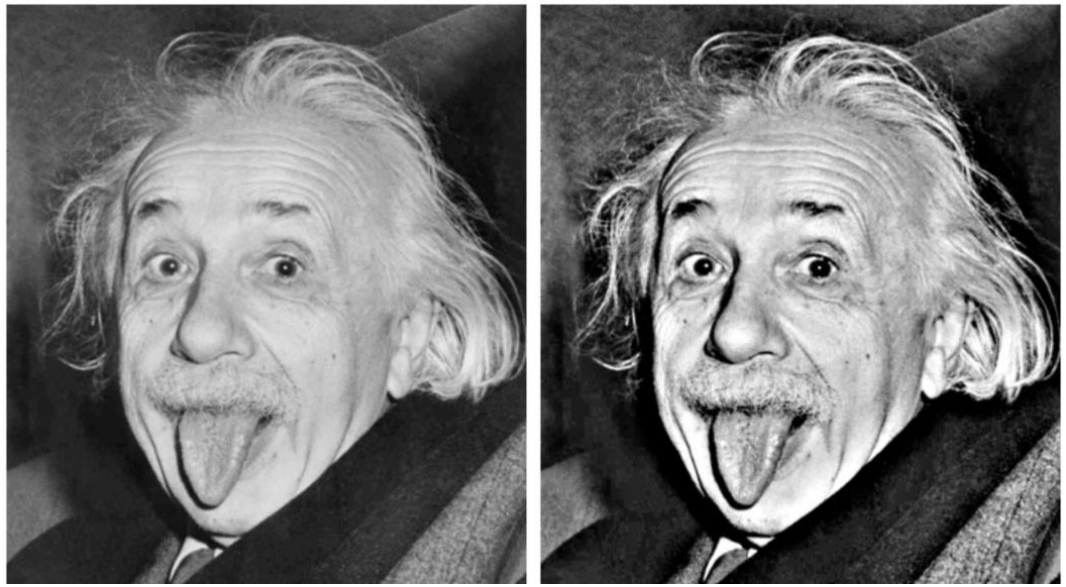
removal of small details, noise reduction

goal of sharpening filters:

highlight intensity transitions (e.g., edges)

averaging analogous to integration

→ sharpening by differentiation



First- and Second-Order Derivatives

partial derivatives in discrete form:

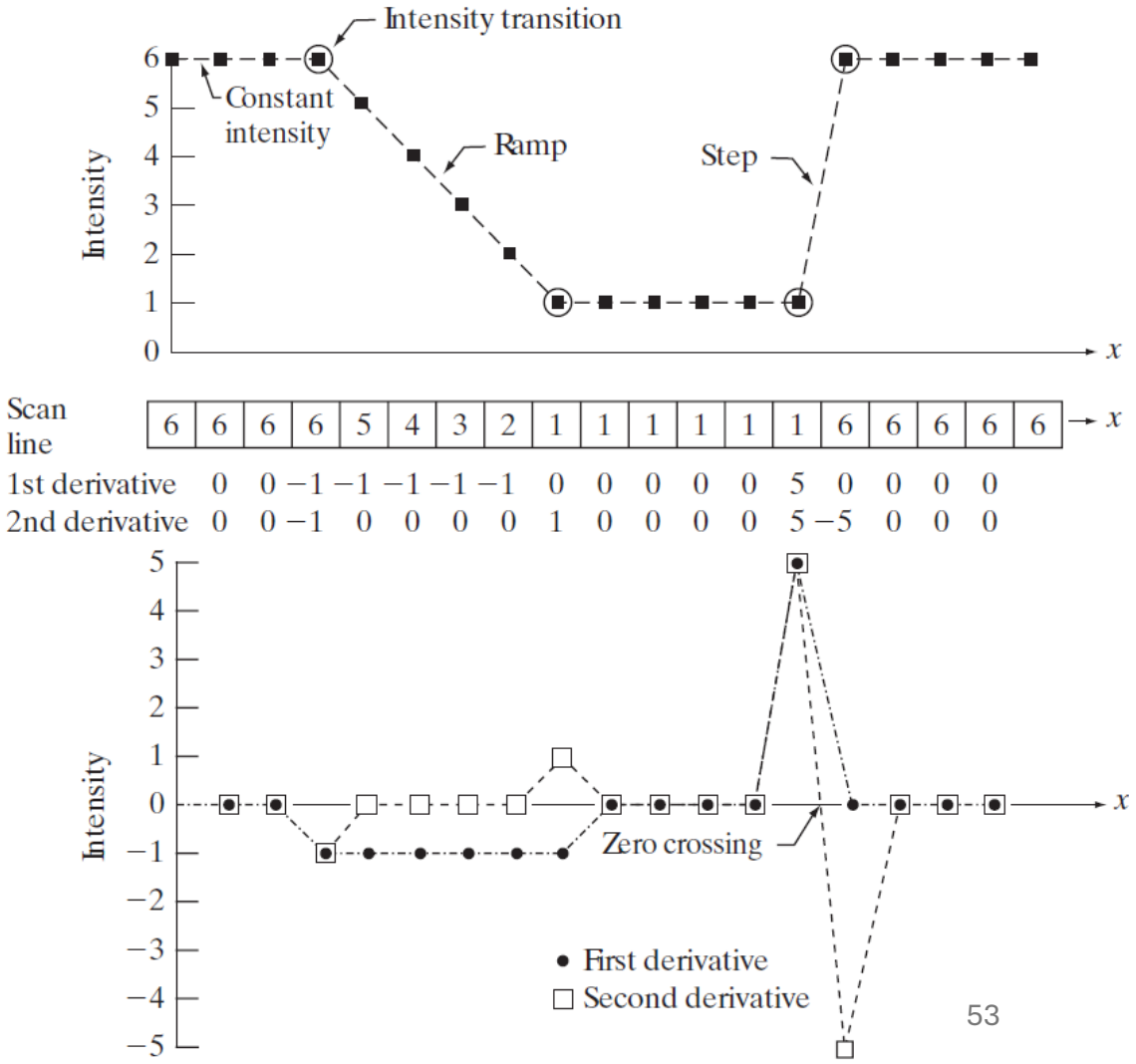
$$\frac{\partial f}{\partial x} = f(x+1, y) - f(x, y)$$

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

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

1D visualization:



Approximated Image Gradient

alternative formulation (1D): $\frac{\partial f}{\partial x} = \frac{f(x+1) - f(x-1)}{2}$

→ 1D derivative filter:

1	0	-1

Sobel filters:

smoothing

1
2
1

$*$ horizontal

-1	0	1

$=$

1	0	-1
2	0	-2
1	0	-1

$\cdot \frac{1}{8}$

construct separable filters:

(convolution kernels)

vertical

-1
0
1

$*$ smoothing

1	2	1

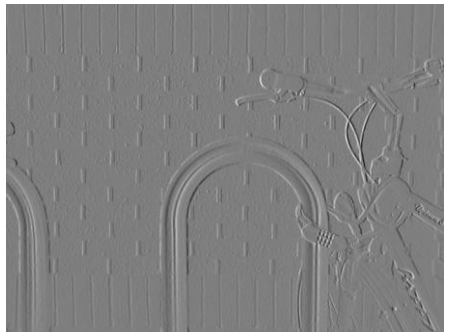
$=$

1	2	1
0	0	0
-1	-2	-1

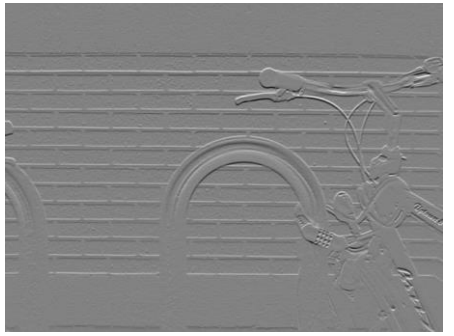
$\cdot \frac{1}{8}$

→ gradient with smoothing

convolved with an image:



$\frac{\partial f}{\partial x}$



$\frac{\partial f}{\partial y}$

Magnitude of Image Gradient

for sharpening:
add this to original

magnitude/amplitude of gradient:

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

magnitude is isotropic

Different Gradient Filters

Gradient	Roberts	Prewitt	Sobel (3x3)	Sobel (5x5)
$\frac{\partial I}{\partial x}$	0 1 -1 0	-1 0 1 -1 0 1 -1 0 1	-1 0 1 -2 0 2 -1 0 1	-1 -2 0 2 1 -2 -3 0 3 2 -3 -5 0 5 3 -2 -3 0 3 2 -1 -2 0 2 1
$\frac{\partial I}{\partial y}$	1 0 0 -1	1 1 1 0 0 0 -1 -1 -1	1 2 1 0 0 0 -1 -2 -1	1 2 3 2 1 2 3 5 3 2 0 0 0 0 0 -2 -3 -5 -3 -2 -1 -2 -3 -2 -1

Good Localization
Noise Sensitive
Poor Detection

→

Poor Localization
Less Noise Sensitive
Good Detection

the larger ones add smoothing components (Sobel: convolution with bilinear filter)

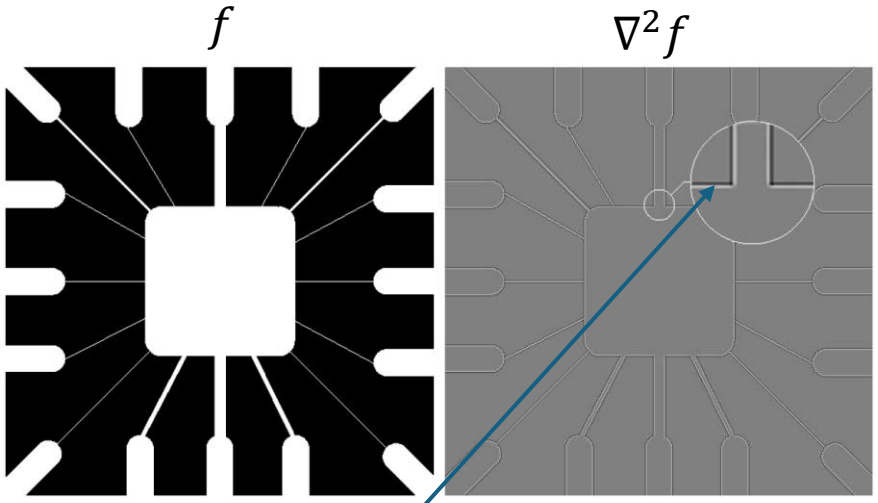
Alternative: Laplacian

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

filter mask:

0	1	0
1	-4	1
0	1	0

$$= f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$



isotropic for rotations in increments of 90°

to yield isotropic results for rotations in increments of 45°:

1	1	1
1	-8	1
1	1	1

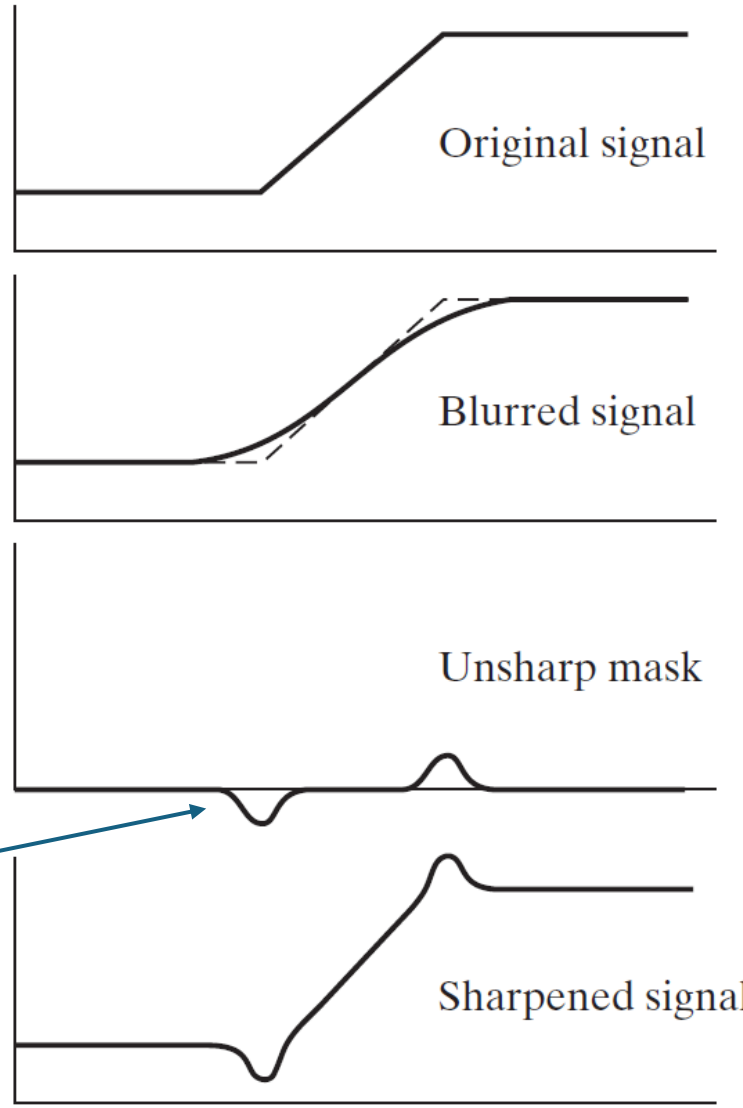
again, add to (or rather subtract from when using this filter definition here) original to get sharpened image:

$$g(x, y) = f(x, y) - \nabla^2 f(x, y)$$

Another Alternative: Unsharp Masking

use of smoothing filters for sharpening

1. smoothing of original image (e.g.,by Gaussian filter)
2. subtract the blurred image from the original → unsharp mask (similar to second-order derivative)
3. add unsharp mask to original image → sharpened image



similar to second-order derivative


original

−


smoothed (3x3)

=


Detailed


original

+


Detailed

=

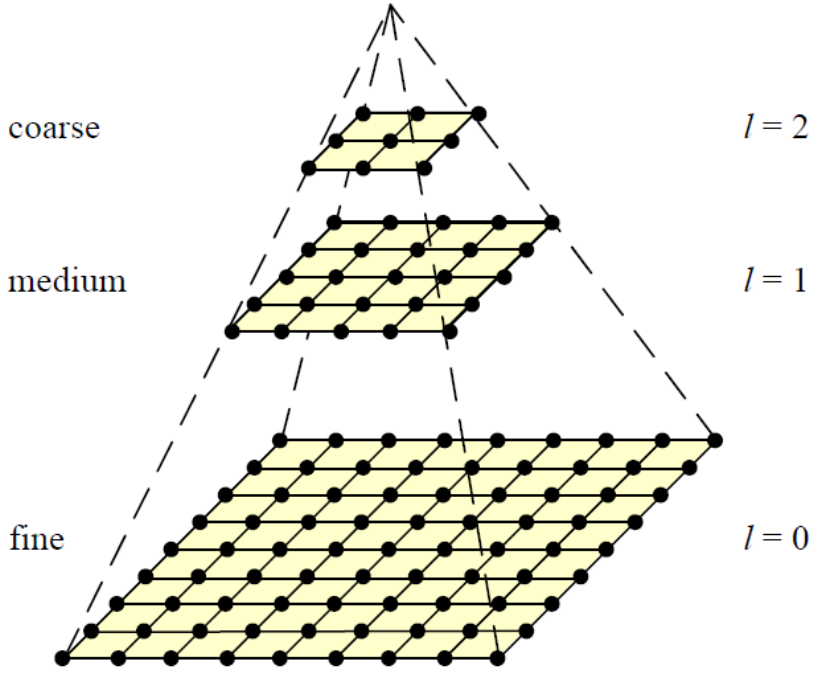

Sharpened

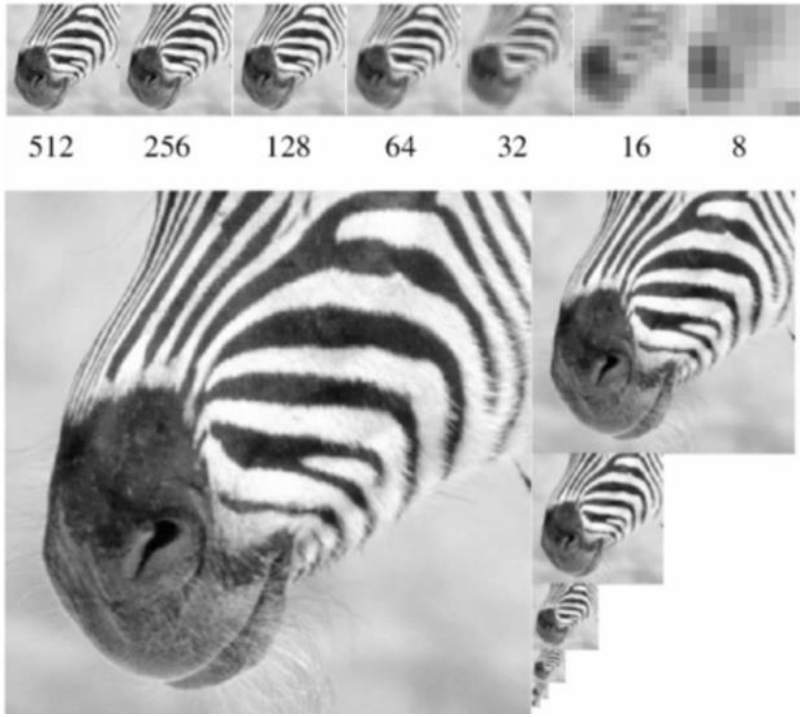
Image Pyramids

Gaussian Image Pyramid

sequence of sub-sampled images (one use case: features at different scales)

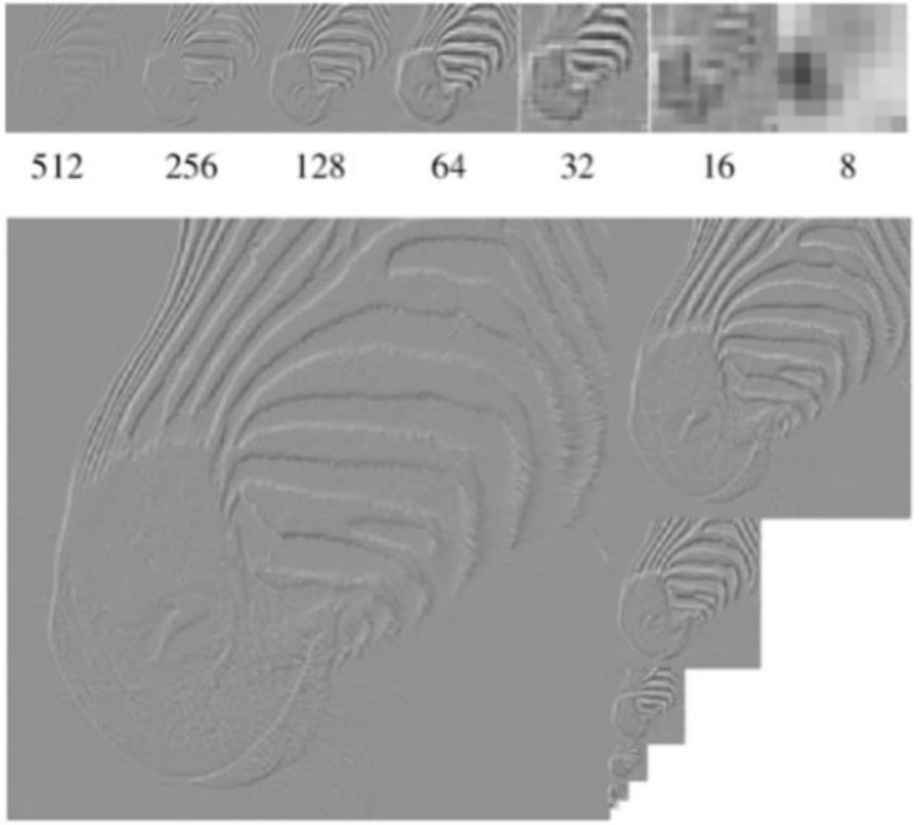
apply smoothing before each down-sample step (to avoid aliasing), typically by convolution with binomial kernel (converging to Gaussian)





Laplacian Image Pyramid

retain only residuals at different levels (allows later upsampling, starting from highest-level image)



difference of Gaussians approximate Laplacian: