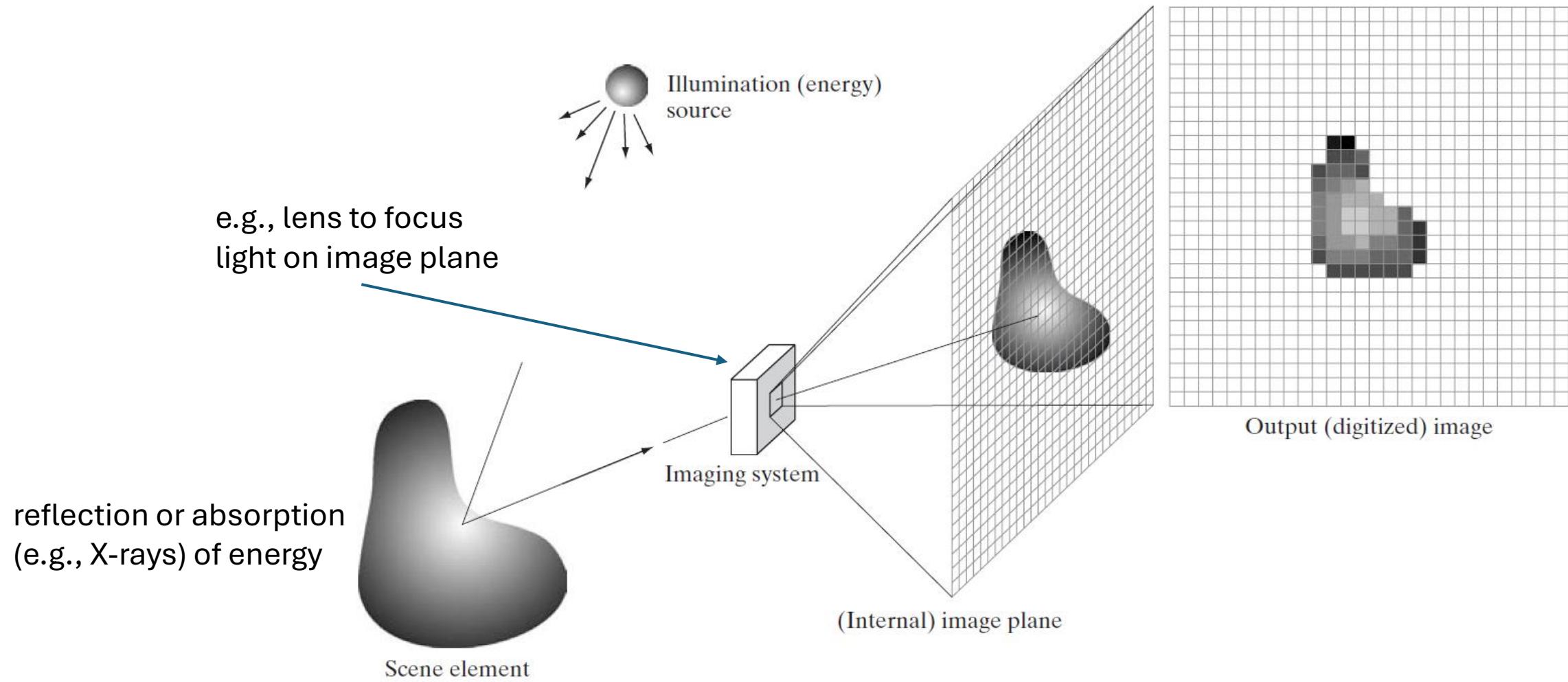


Digital Image Processing

Computer Vision

Digital Image Acquisition Process



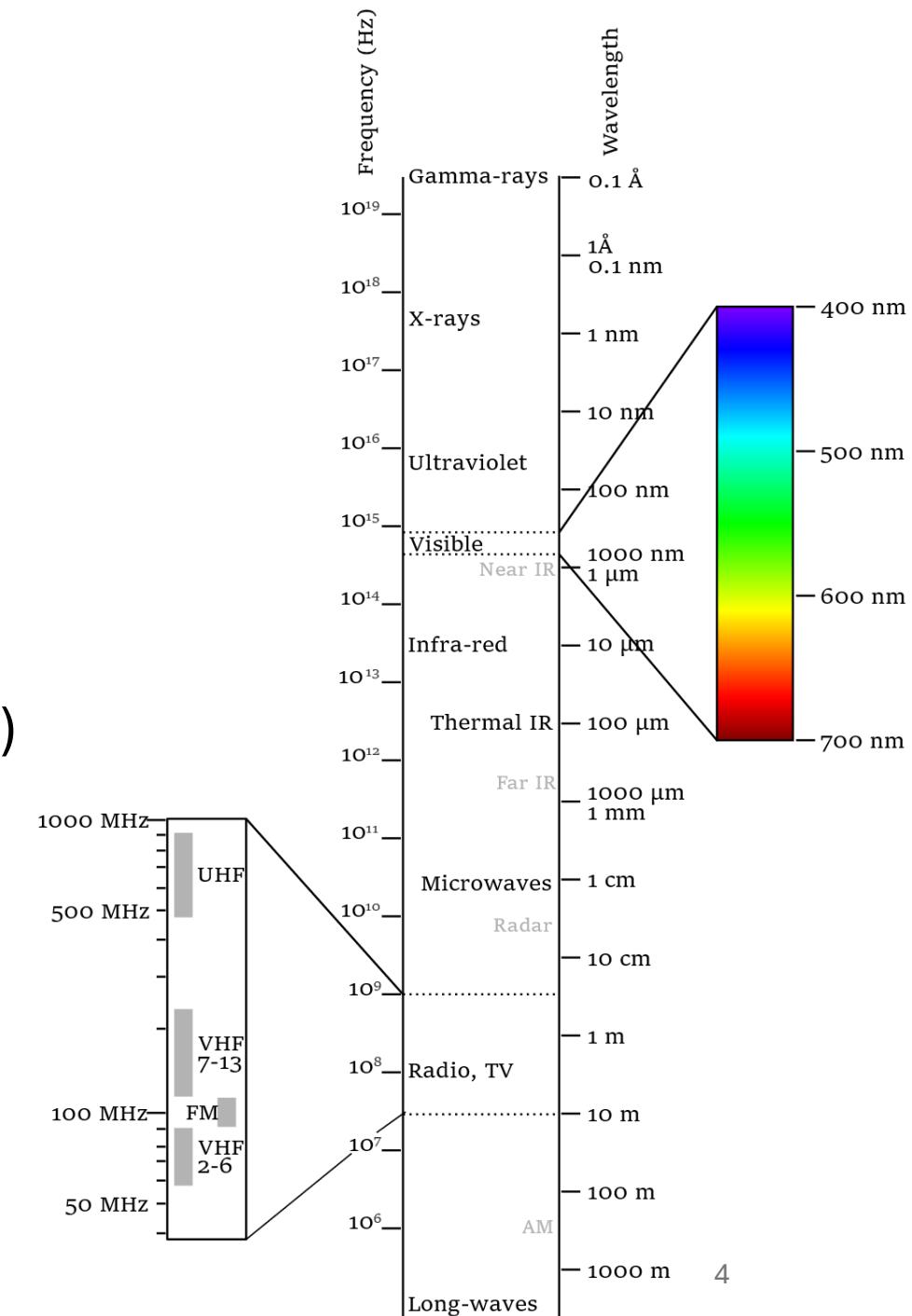
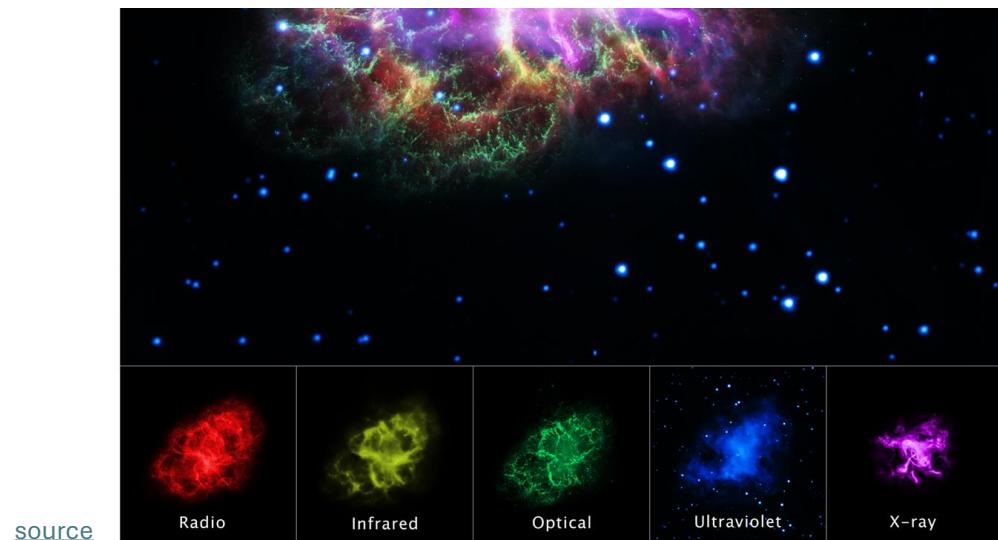
Energy Sources

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



Lidar vs Cameras for Self-Driving Cars

Lidar:

- similar to radar, but using lasers
- very accurate



camera:

- human-like perception
- cost-effective



Examples for other Energy Sources

ultrasound



electron beams
(electron microscopy)

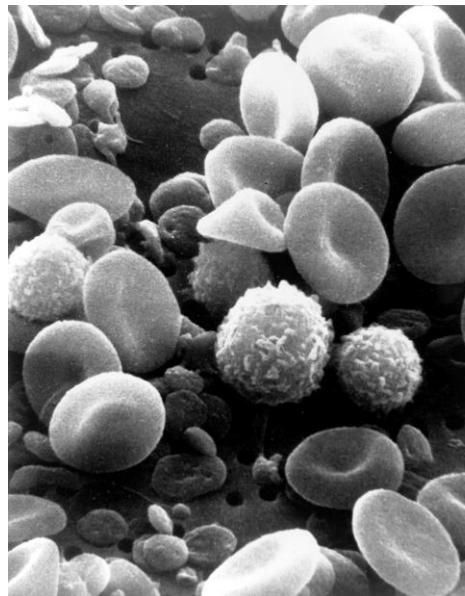
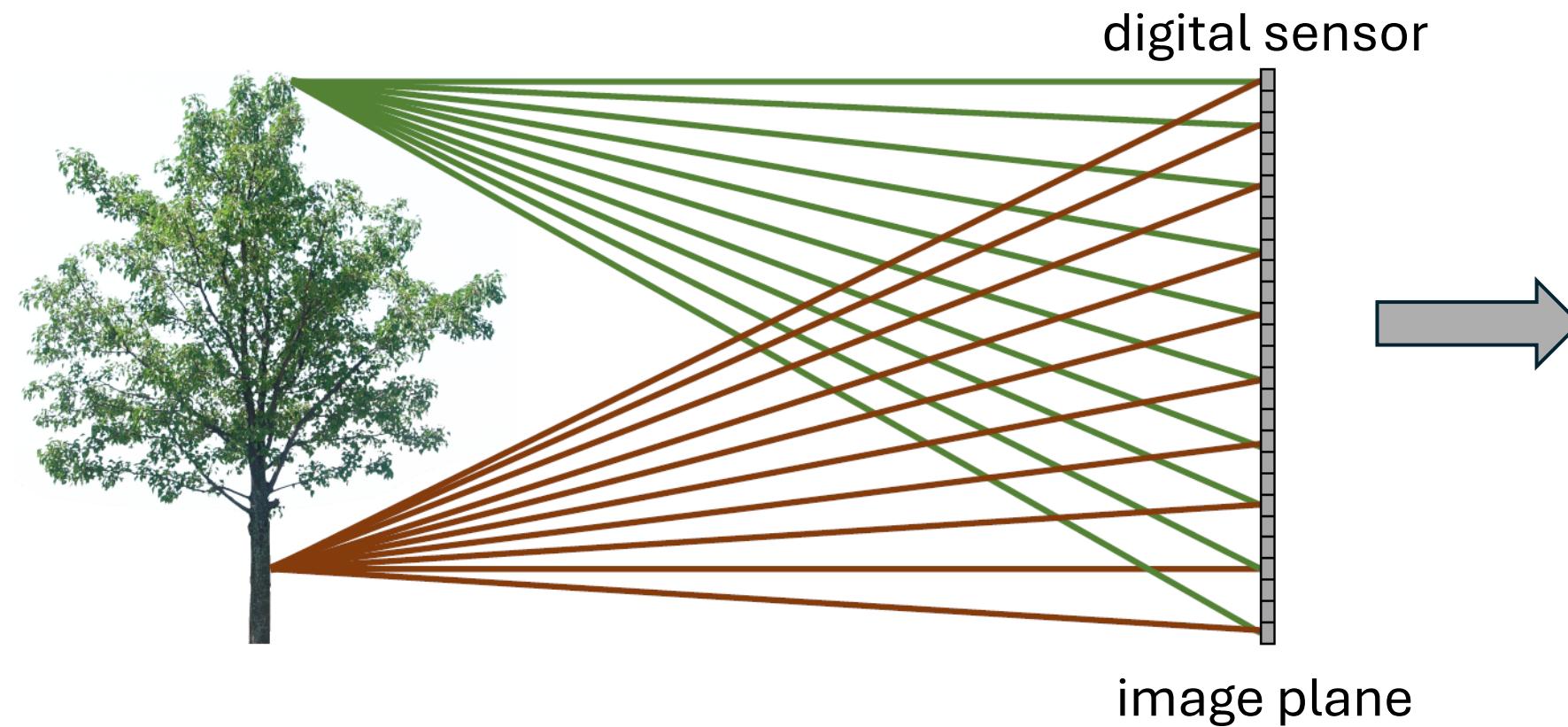
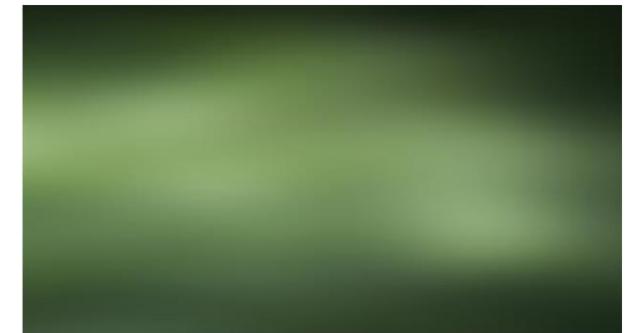


Image Formation

Bare-Sensor Imaging

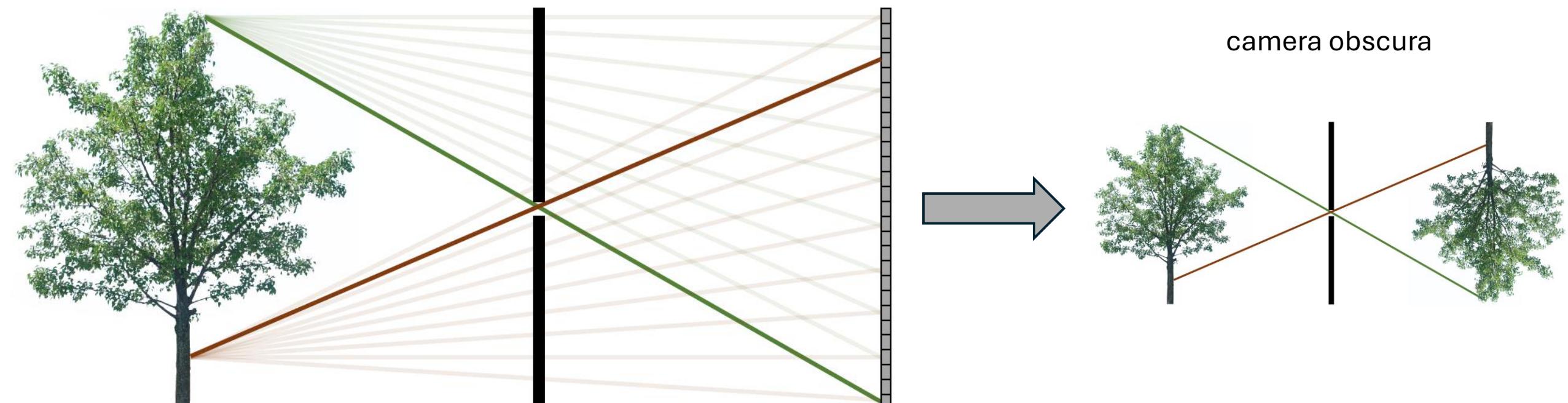


all scene points contribute
to all sensor pixels:



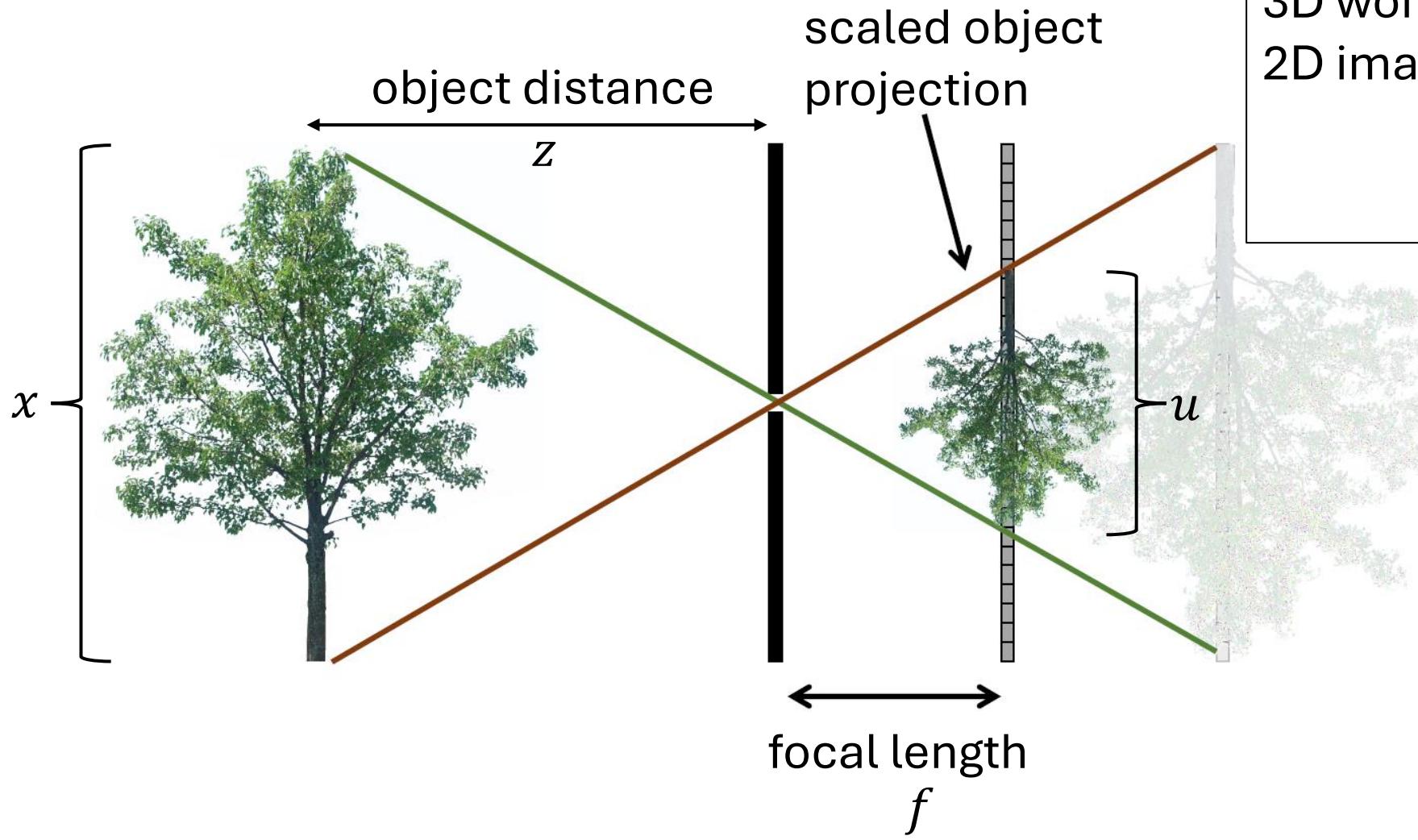
Pinhole Camera

camera: barrier with
pinhole (aperture)



each scene contributes
to one pixel only

Pinhole Camera Model



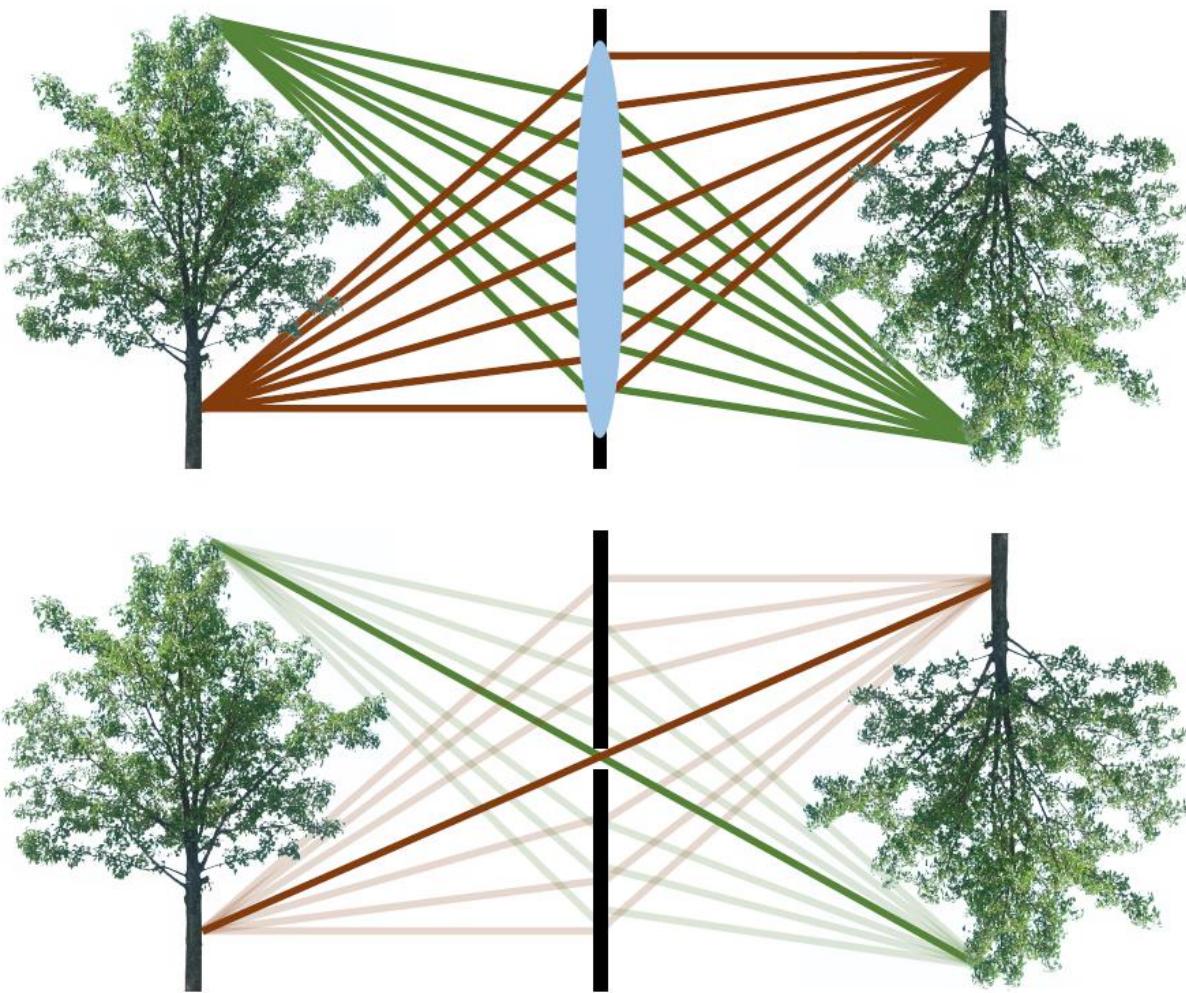
3D world coordinates (x, y, z) to
2D image coordinates (u, v) :

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

losing depth ...



Lens Camera

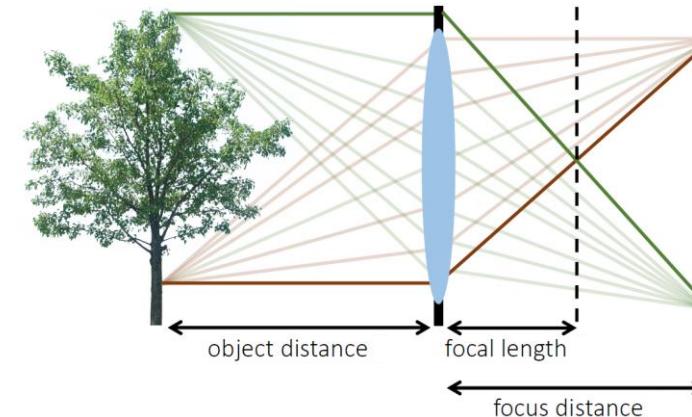


ideal pinhole is infinitesimally small: larger diameter/aperture makes image blurrier (but also allows more light to pass through: higher intensity of image pixels)

→ 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 focus length of pinhole camera)



Digital Image Sensors

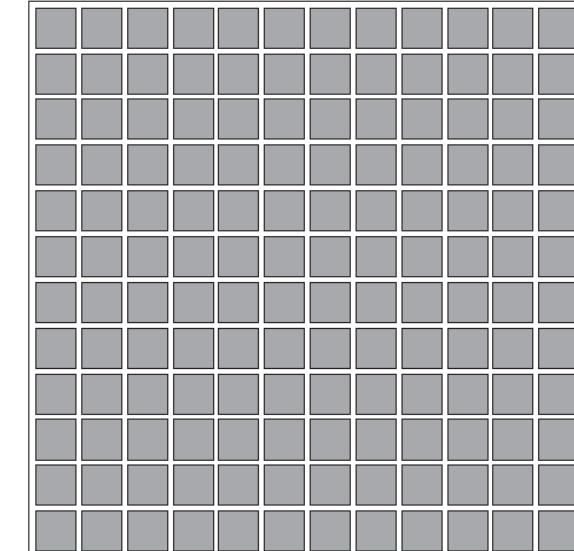
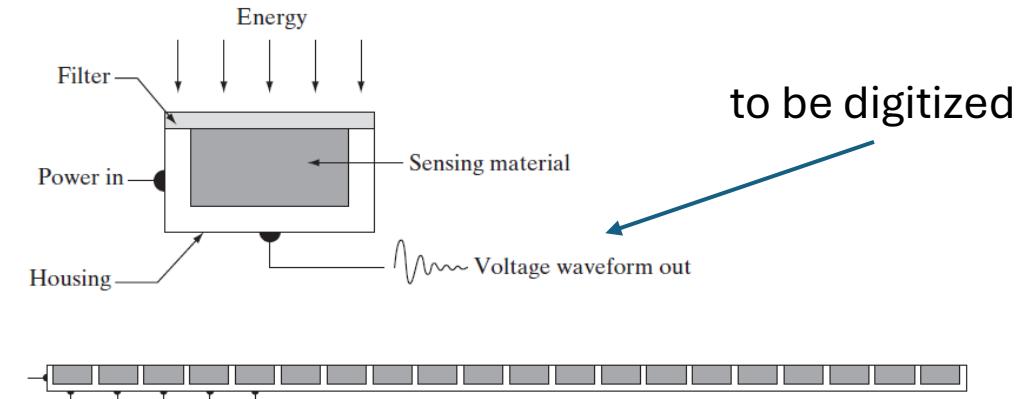
Photons to Electrons

largely replaced chemical and analog imaging

single sensor: photodiode

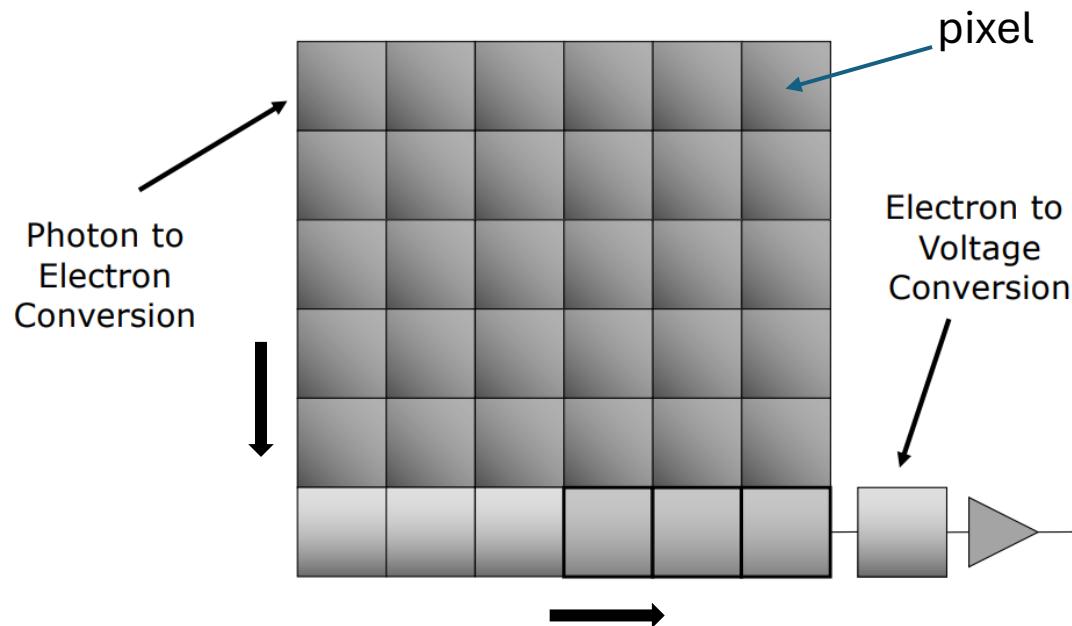
sensor strips: motion perpendicular to strip for imaging in other direction, used in CT

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

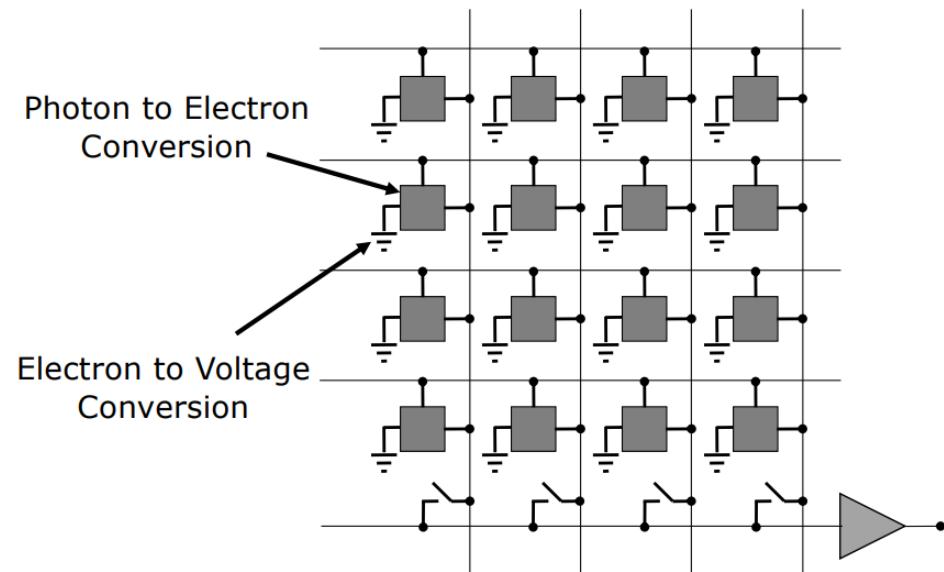
CCD: Charge Coupled Device



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

active pixel sensor

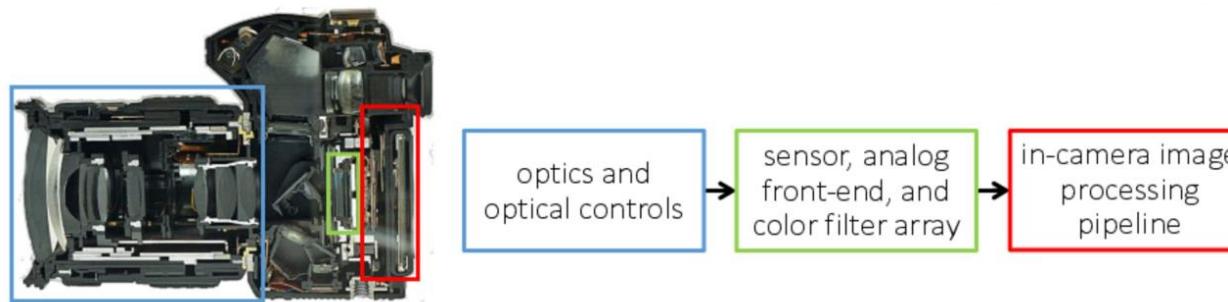
CMOS: Complementary Metal-Oxide Semiconductor



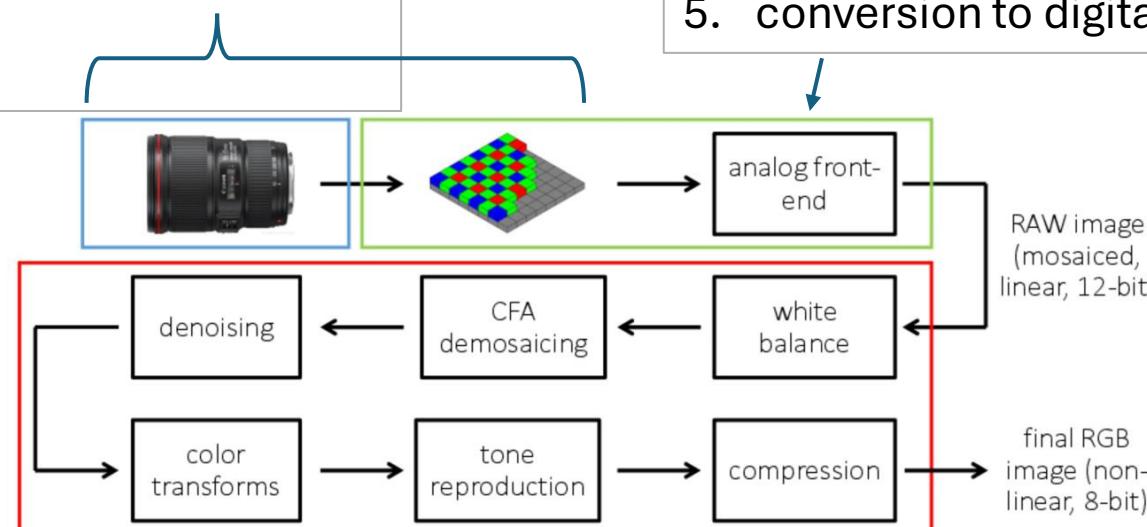
lower power consumption, typically smaller
→ used in most consumer product cameras

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

The modern photography pipeline

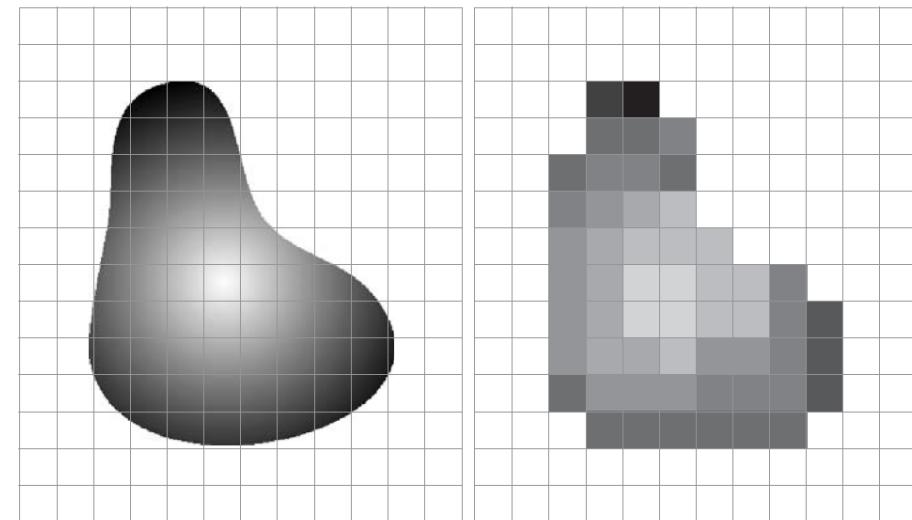
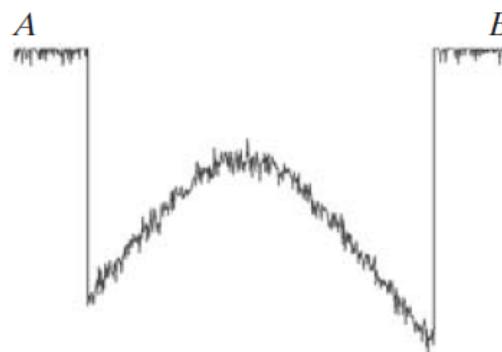
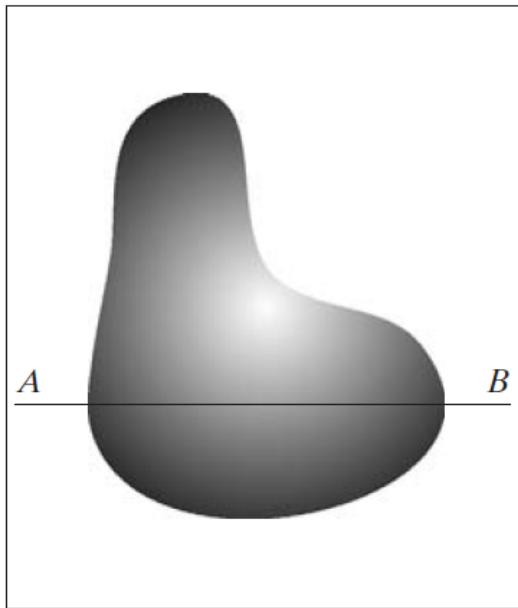


- 1. camera shutter opens
 - 2. photodiodes: photons to electrons, store in potential well
 - 3. camera shutter closes
- 4. read-out of potential wells, conversion to analog signals
 - 5. conversion to digital signals (intensity levels)

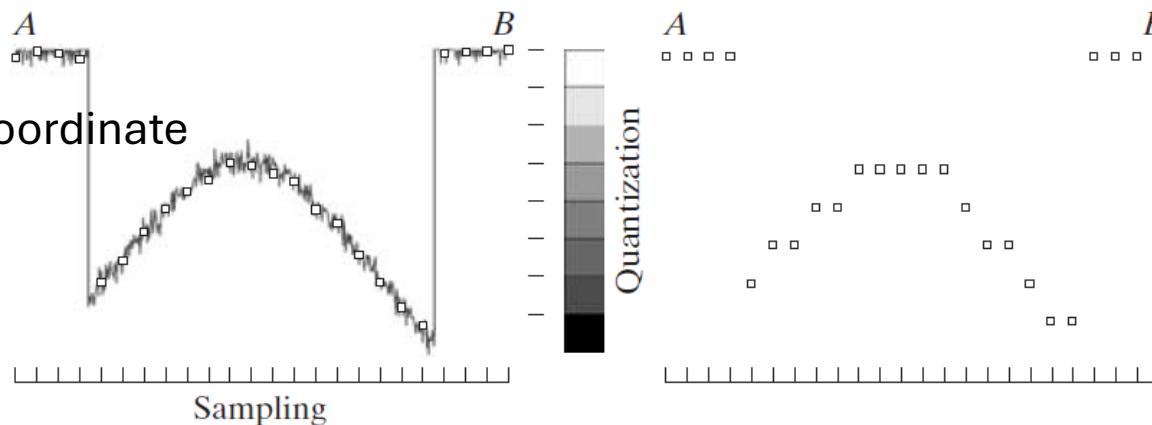


Pixels & Intensity

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)

0	2	15	0	0	11	10	0	0	0	9	9	0	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

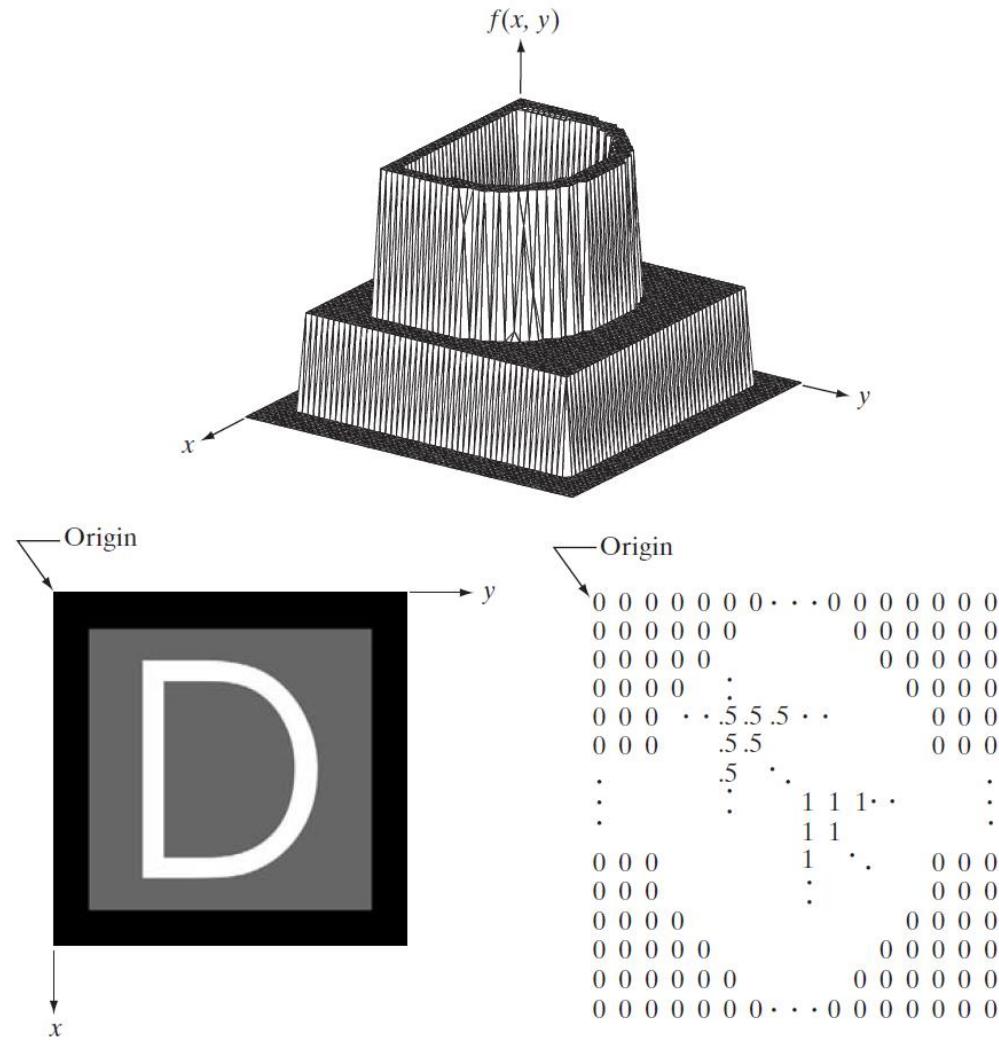
[source](#)

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

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

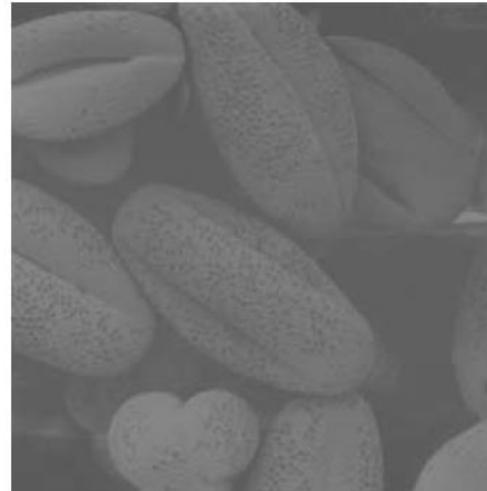


Contrast

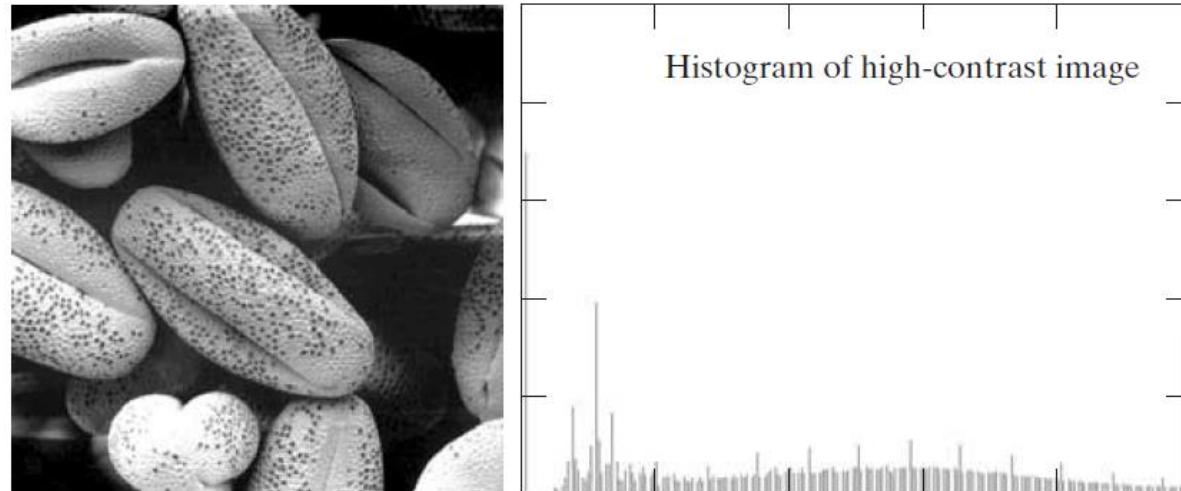
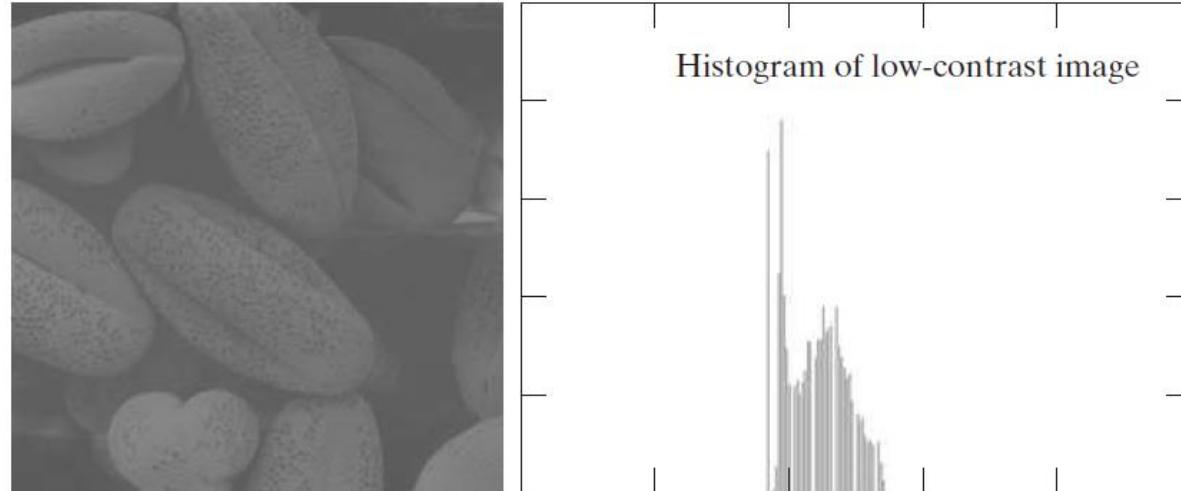
range of intensity levels:

- 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:

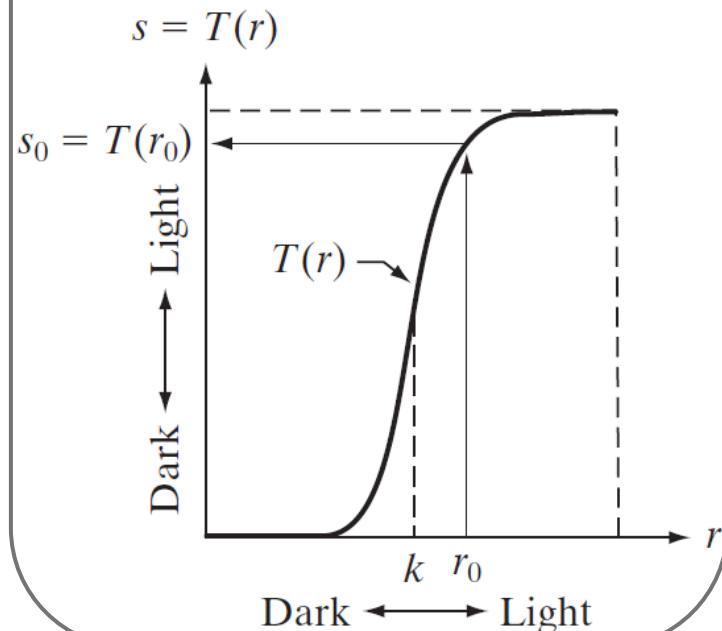


lighten or darken image simply by adding or subtracting a fixed value from the pixel intensities

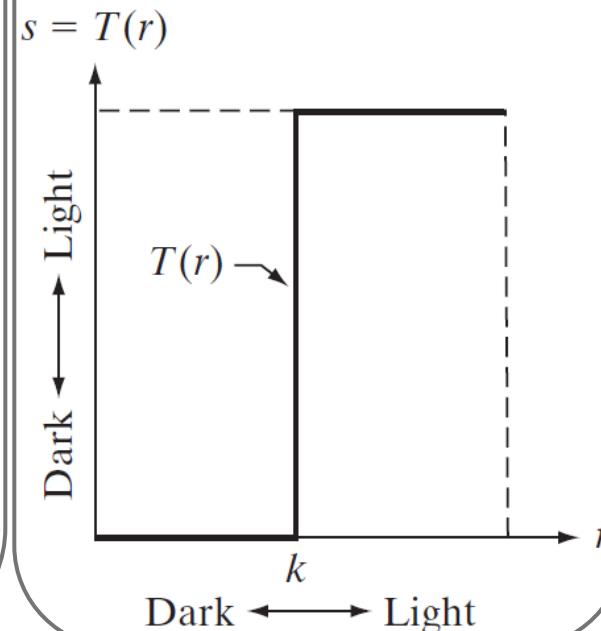
Intensity Transformations

e.g., image enhancement through contrast manipulation

contrast stretching
(expanding range of intensity levels):



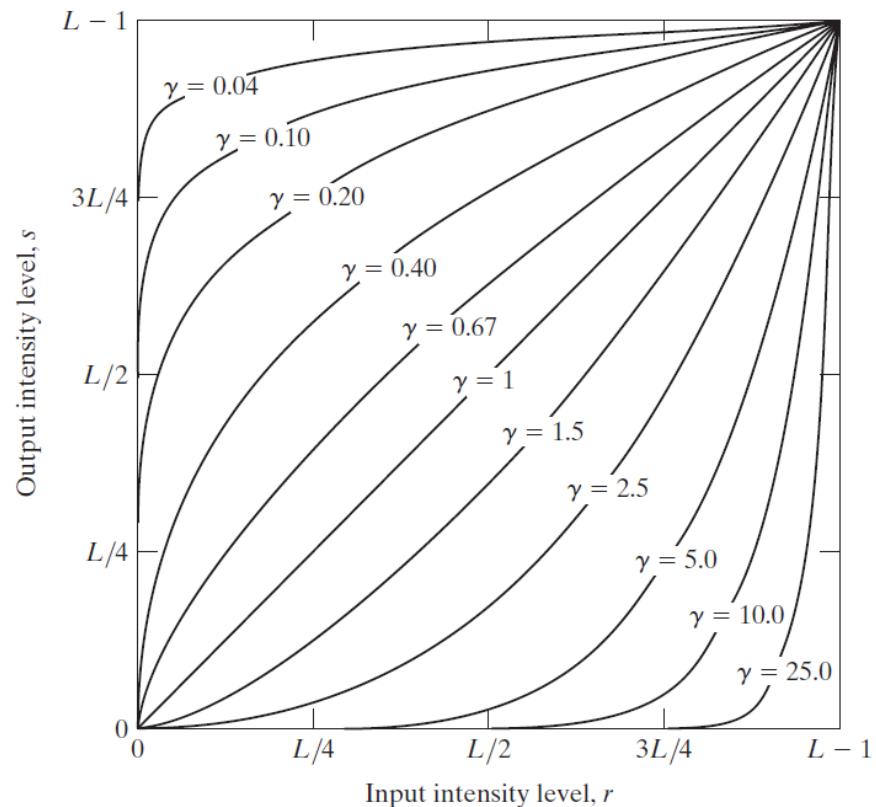
thresholding
(differentiation of foreground and background regions):



Power-Law Transformations

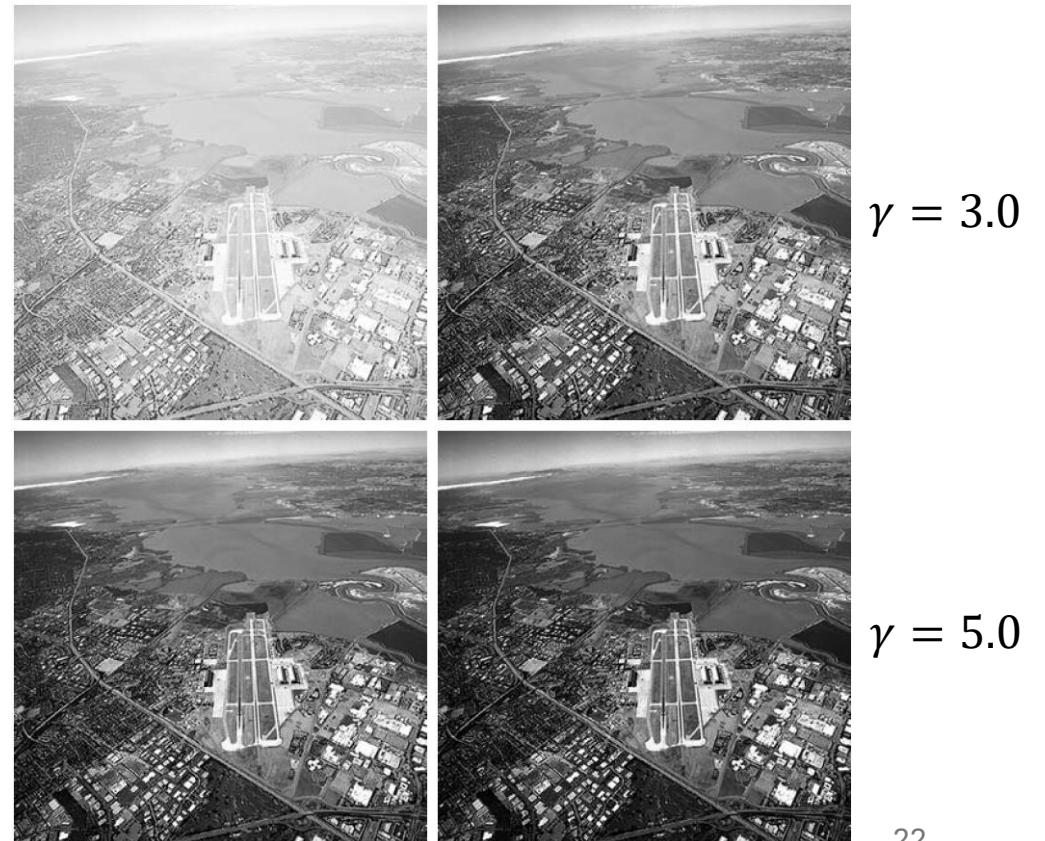
simultaneous spreading and compression of intensity levels

e.g., gamma correction:



$$s = r^\gamma$$

$$\gamma = 4.0$$



Histogram Equalization

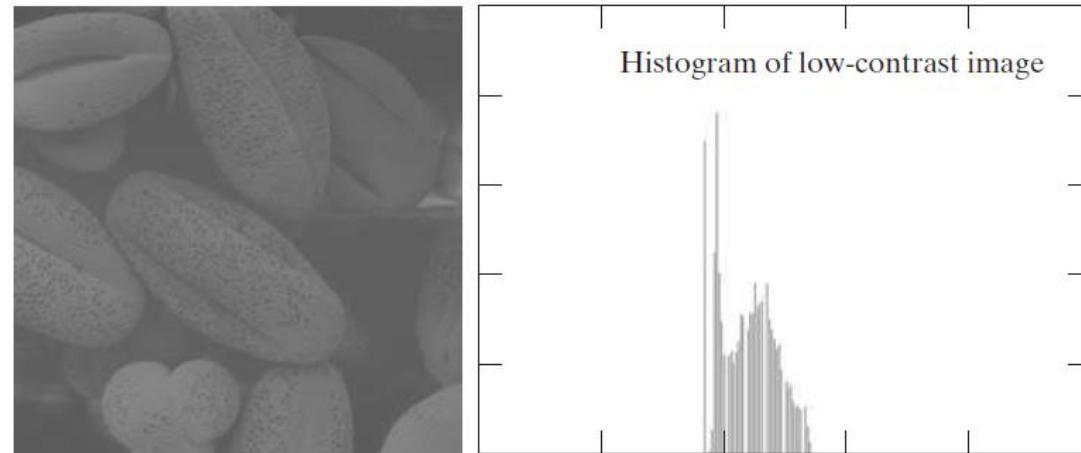
automatic contrast stretching

$$s_k = \frac{(L - 1)}{MN} \sum_{j=0}^k n_j \quad k = 0, 1, 2, \dots, L - 1$$

maximum intensity level
number of pixels in bin k
number of pixels

global transformation function: based on intensity distribution of entire image

local histogram equalization by restricting transformation to pixel neighborhoods



after histogram equalization:

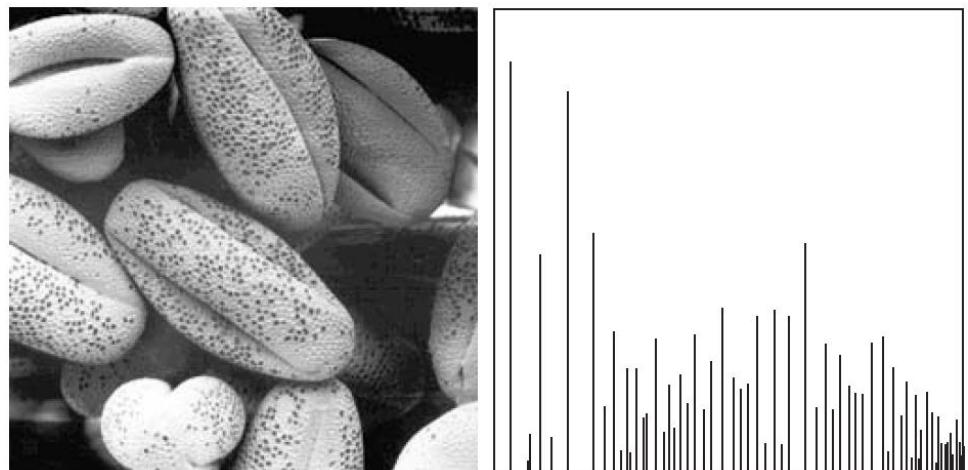
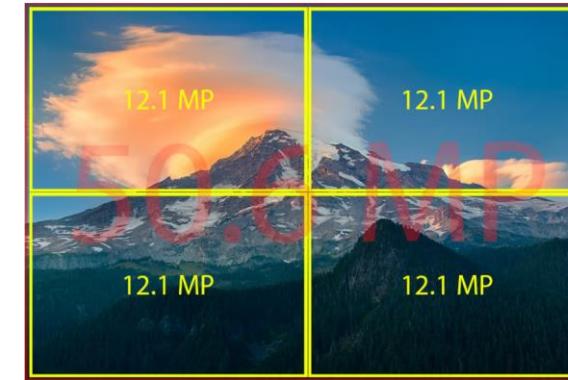


image size: number of pixels

Spatial Resolution

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

→ determines print size of digital image with given pixel count



do not confuse digital image resolution with screen resolution

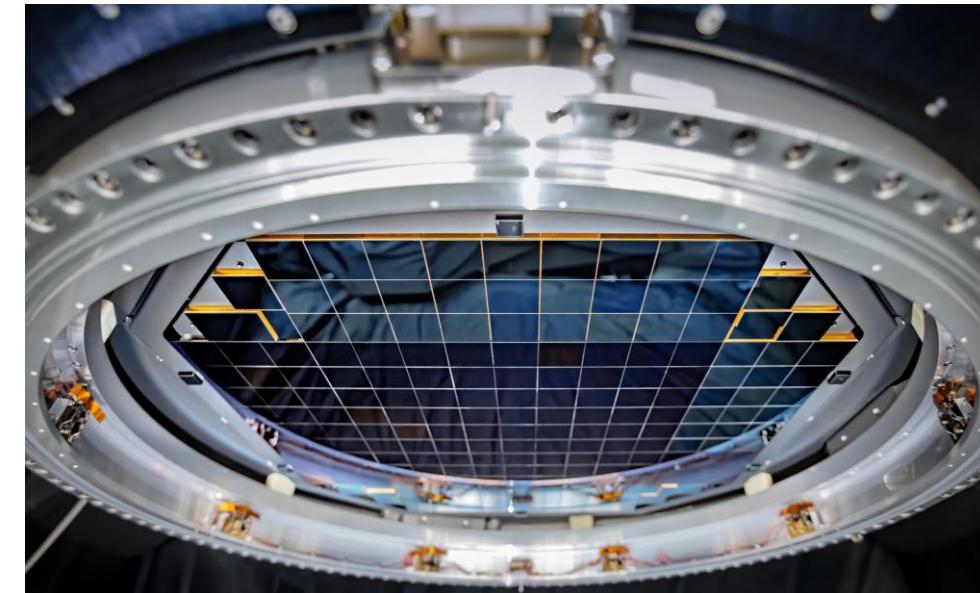
e.g., iPhone 16 Pro Max: 2868×1320 pixels, 6.9 inches diagonal → 460 ppi

also, do not confuse with dots per inch (dpi): property of the printer

Example from Astronomy

Vera C. Rubin Observatory

first light expected in 2025



largest digital camera ever made: 3.2-gigapixel CCD



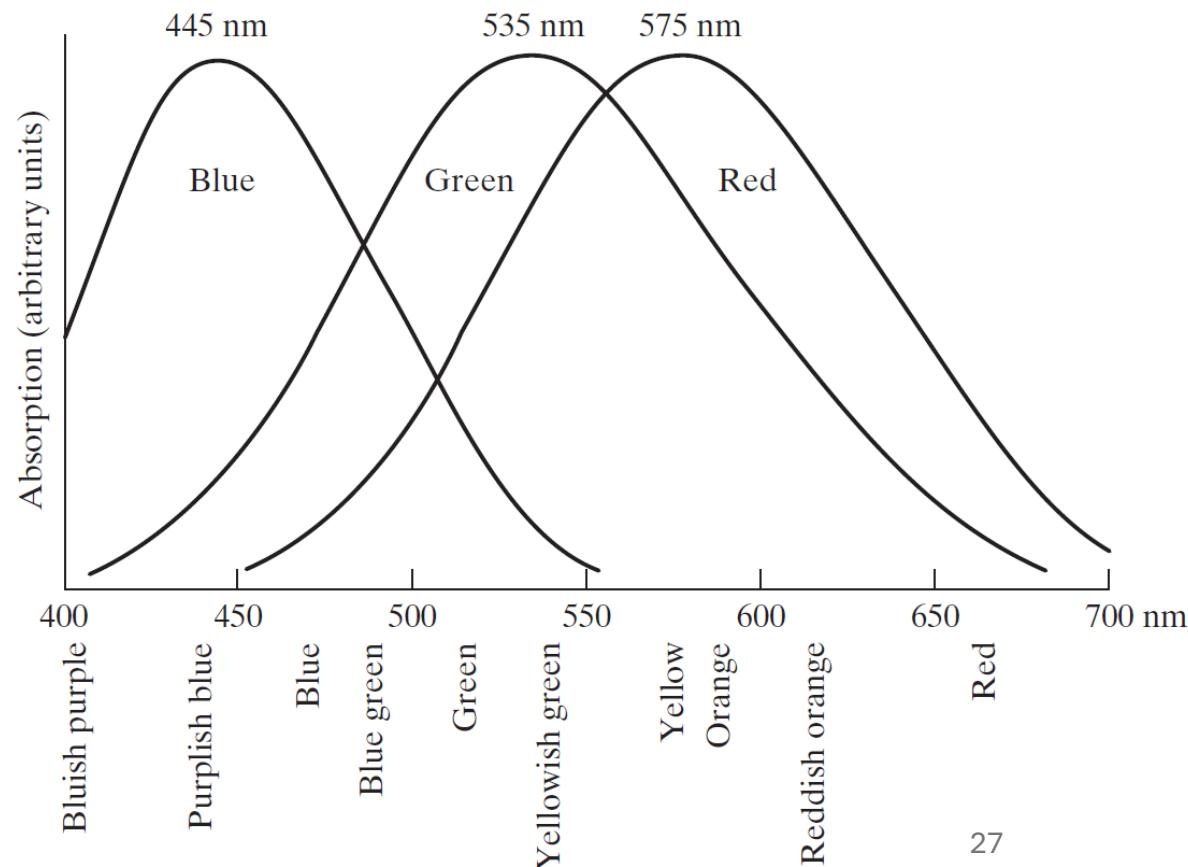
Color

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-sensitive cells)

three types of cones: red, green, blue
→ combinations of red, green, and blue create perceived colors



Primary Colors

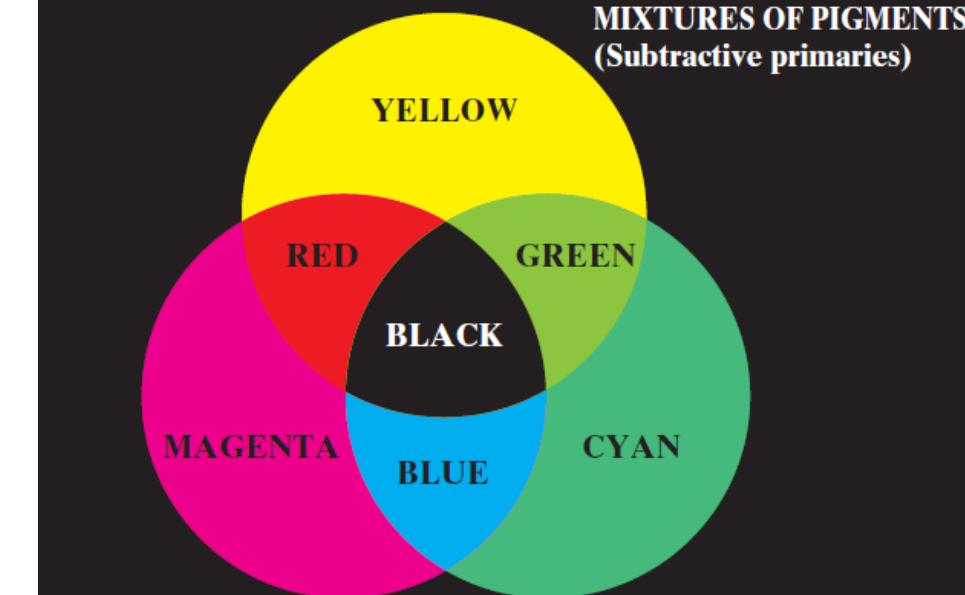
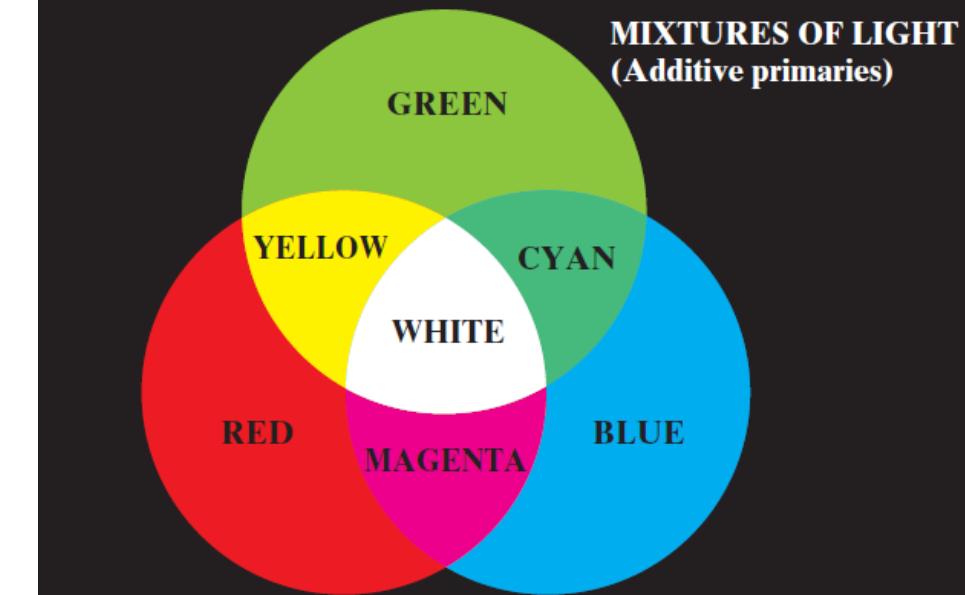
standardization:

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:

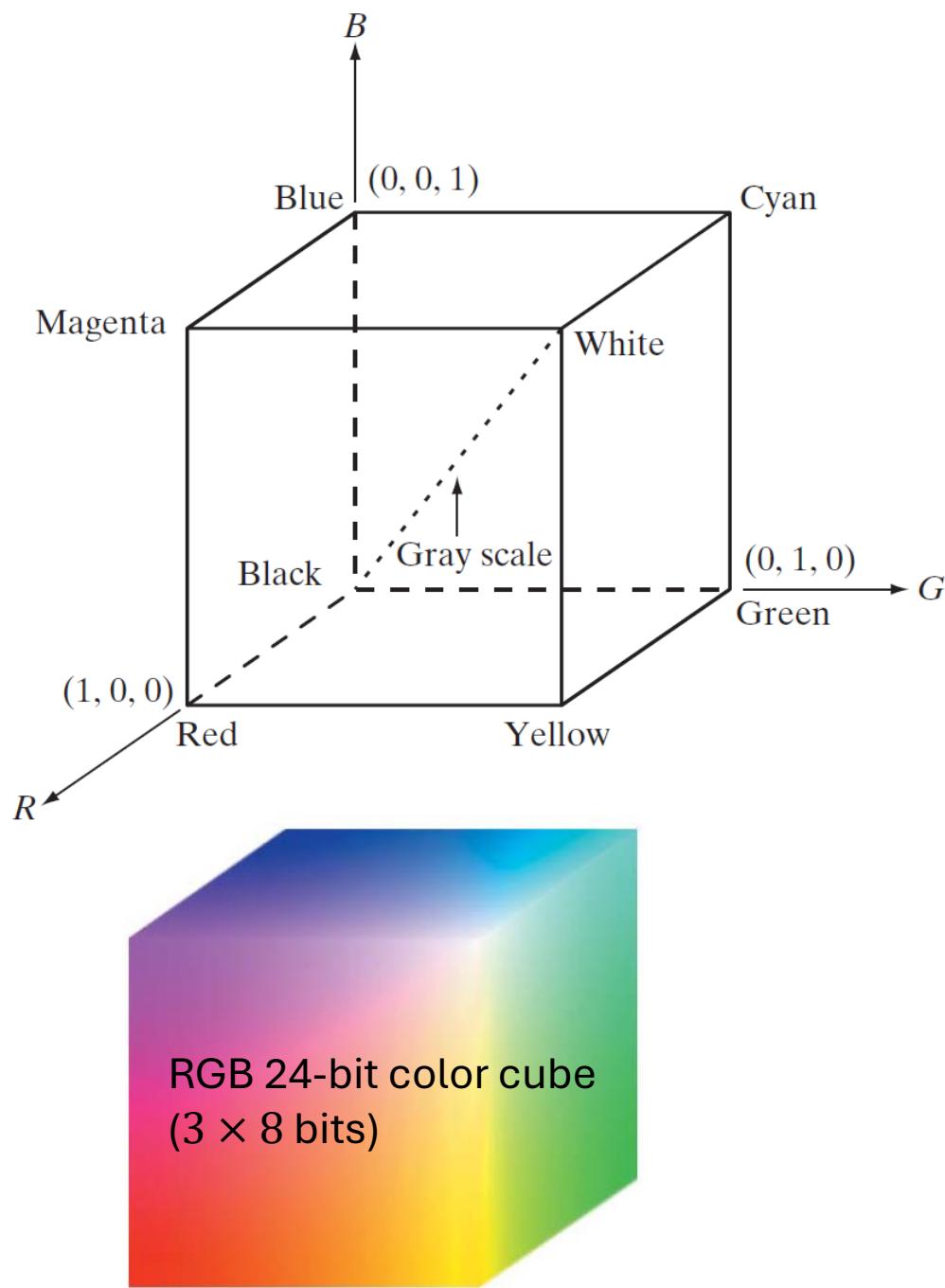
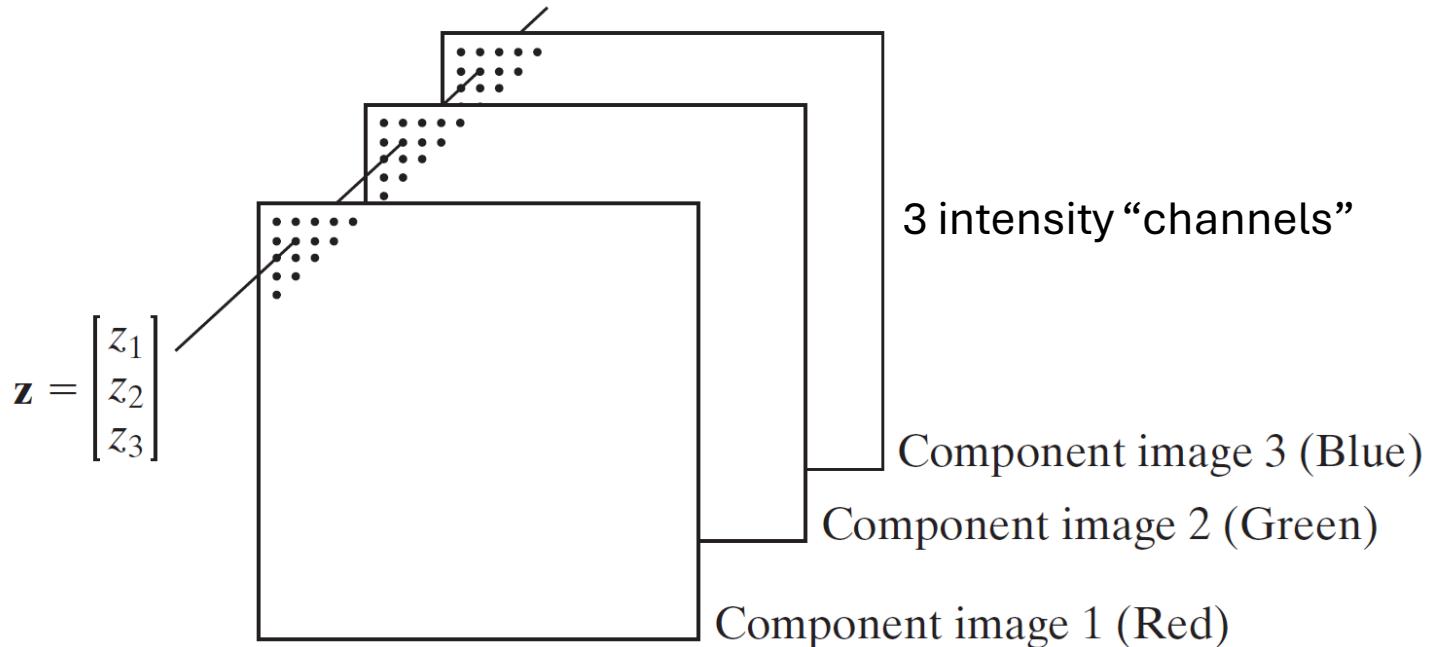
primary colors of pigments each absorb one primary color of light



RGB Color Model

color model for monitors and cameras
(other color models: CMYK, HSV, ...)

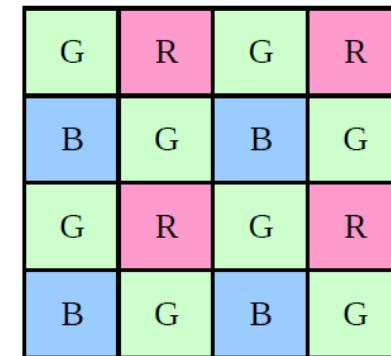
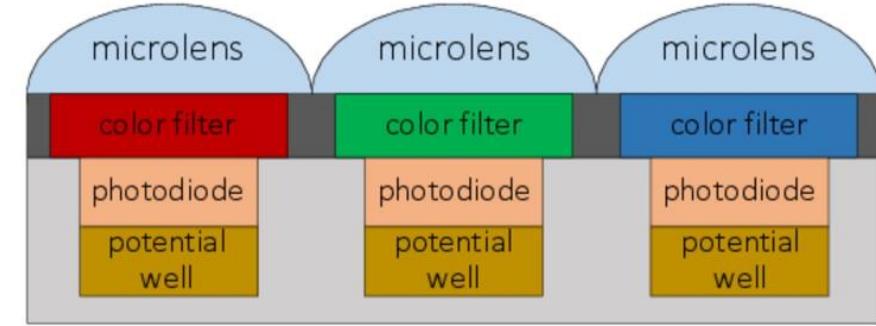
color pixels:



Color Sensing

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

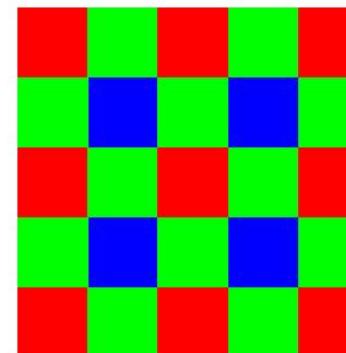
estimation of missing color values for each pixel by interpolation from neighboring pixels
→ full-color image



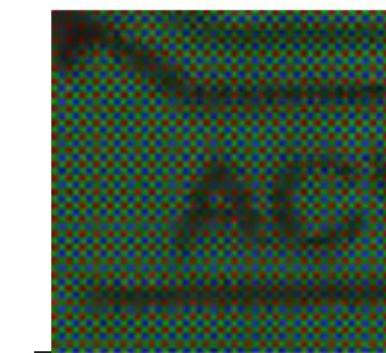
Bayer RGB Pattern

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

Image Compression

Need for Compression

crucial for storage and transmission

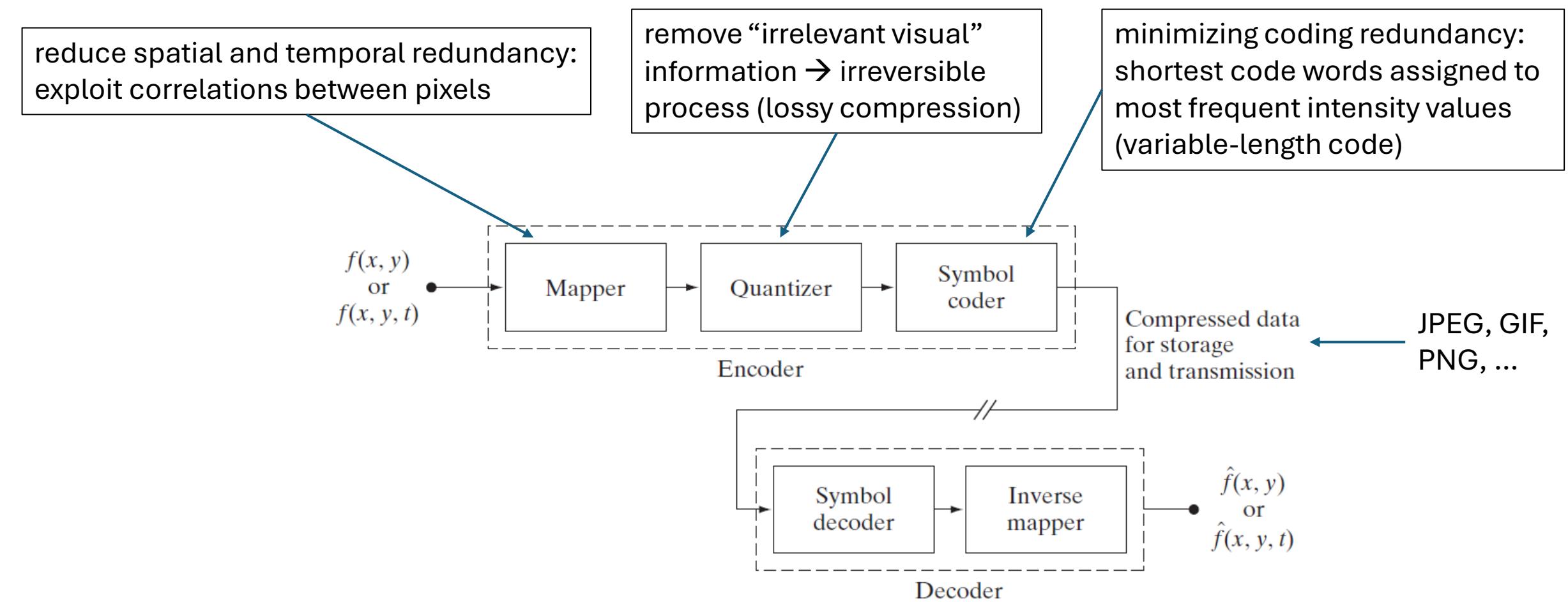
example HD TV:

image resolutions of $1920 \times 1080 \times 24$ bits

digital video as sequence of full-color still images (frames) with 30 frames per second

→ 2-hours video 1.34 TB

Compression & Decompression



Example: Block Transform Coding

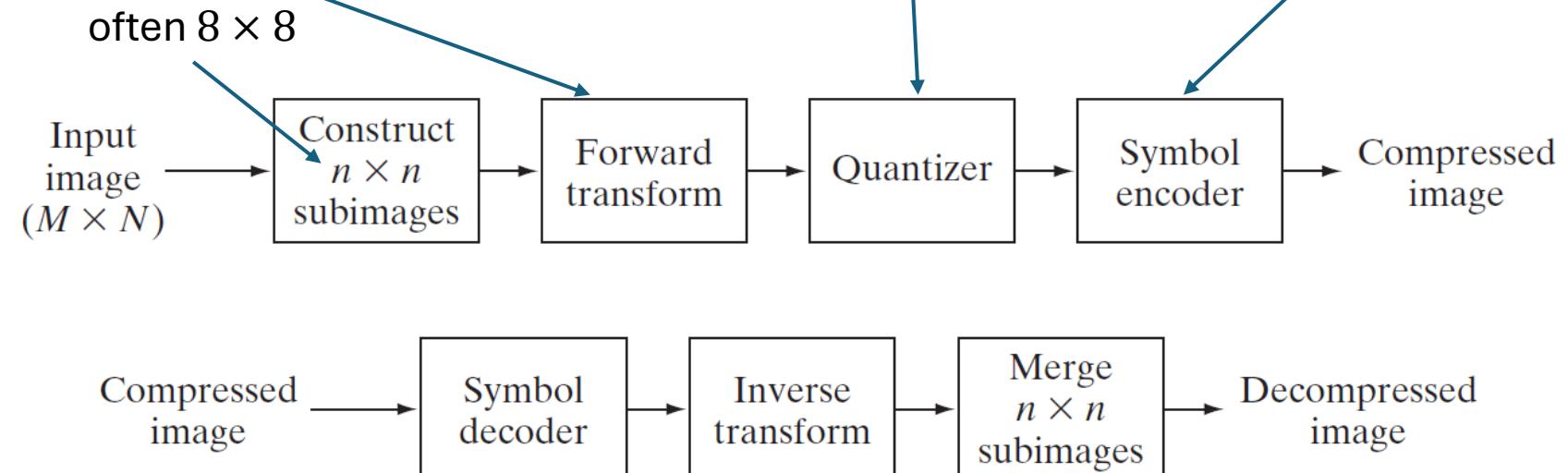
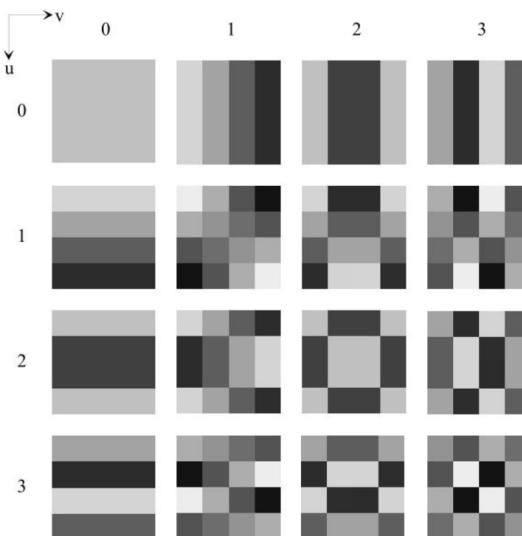
decorrelate pixels of each sub-image by means of discrete 2D transform

- e.g., discrete cosine transform (DCT) or wavelet transform
- DCT kind of approximated principal component analysis (PCA)

drop or coarsely quantize least important transform coefficients

encode quantized coefficients (e.g., Huffman coding)

DCT basis functions:



Neighborhood Operators

Single-Pixel/Point Operation



point processing
intensity transformations

Neighborhood Operation



“filtering”

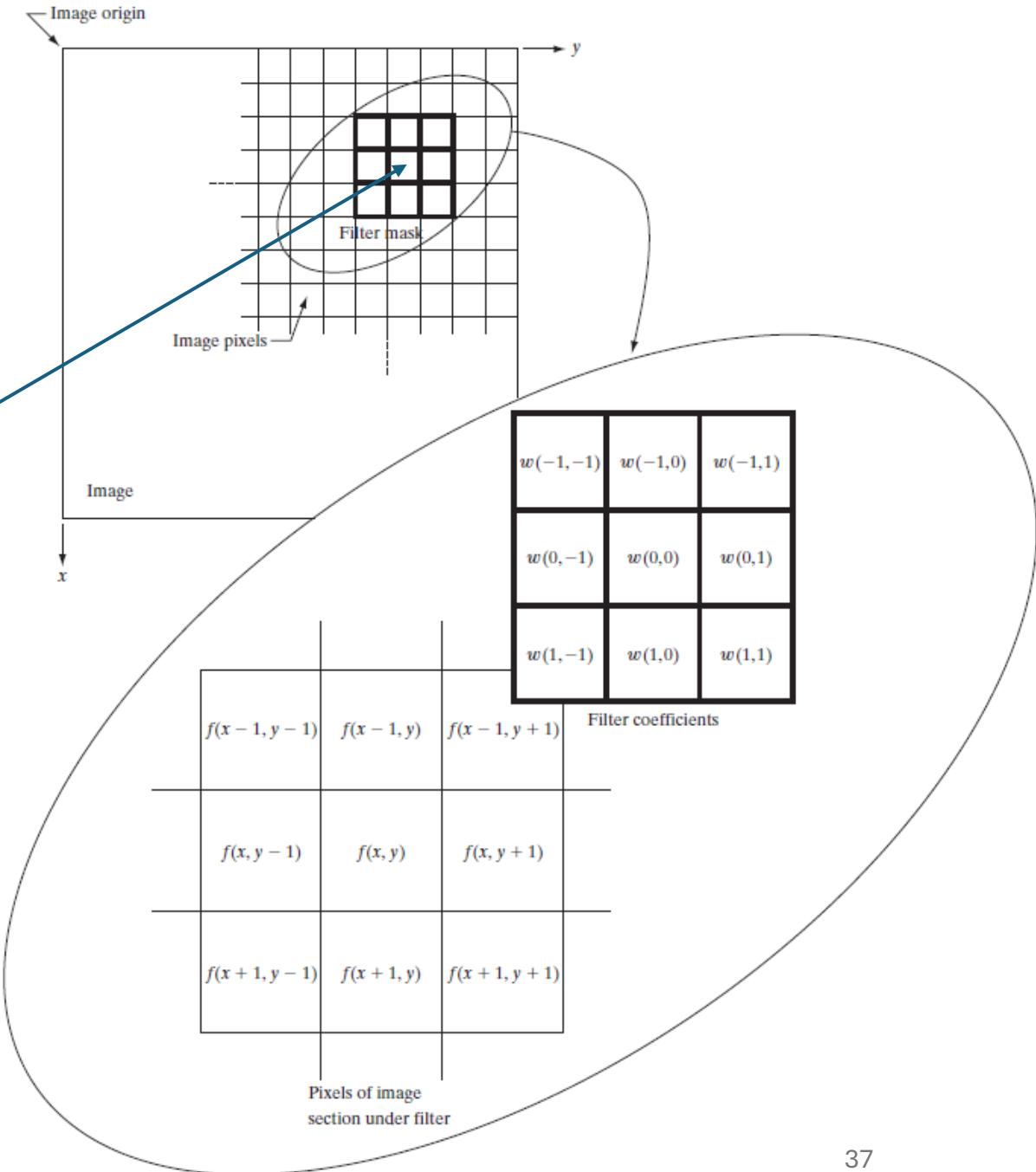
Spatial Filtering

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.

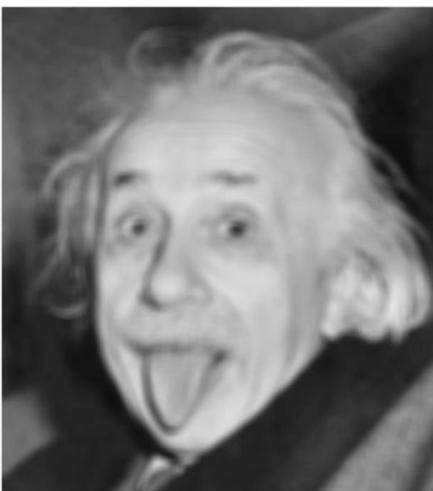
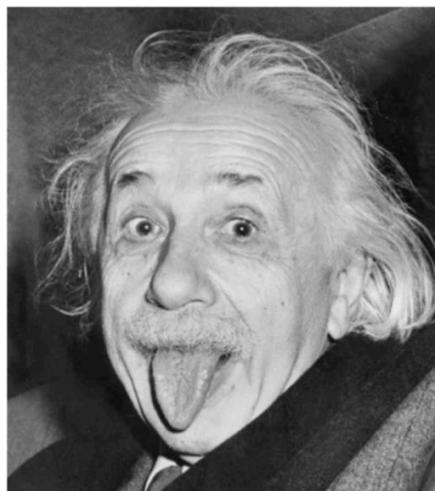
Example: Box Filter

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

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

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)

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:

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{matrix} * \begin{matrix} 1 & 2 & -1 \\ 1 & 1 & -1 \\ -2 & -1 & 0 \end{matrix} = \begin{matrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & -1 & 0 \\ 0 & 1 & 1 & -1 & 0 \\ 0 & -2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{matrix}$$

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):

$\frac{1}{16}$	1	2	1
	2	4	2
	1	2	1

$$\frac{1}{256}$$

Gaussian filter:

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

$\frac{1}{4}$	1	2	1
	1	4	1

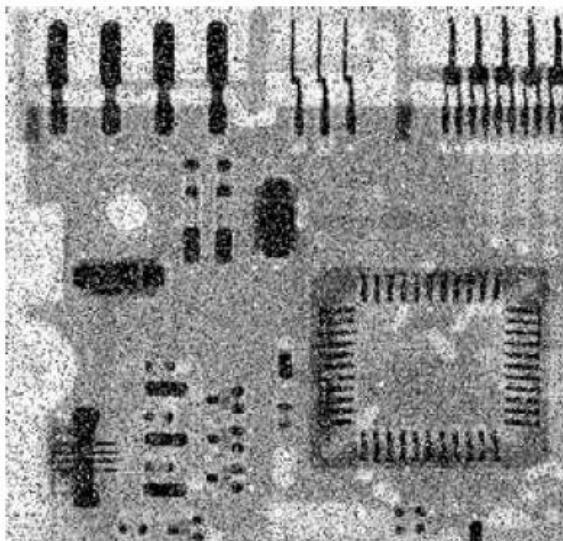
$\frac{1}{16}$	1	4	6	4	1
	1	4	6	4	1

corresponding horizontal 1D kernels

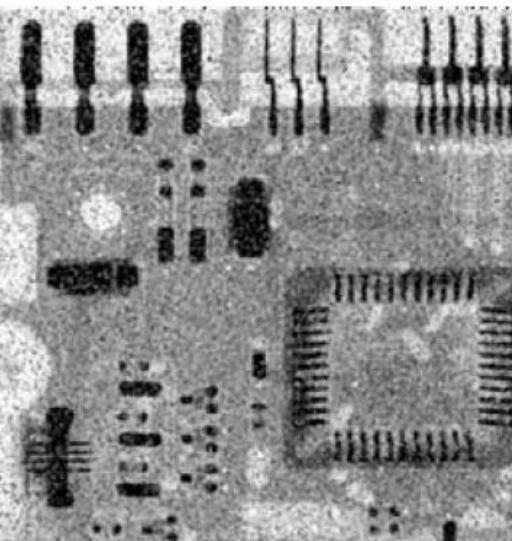
Median Filter

- 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

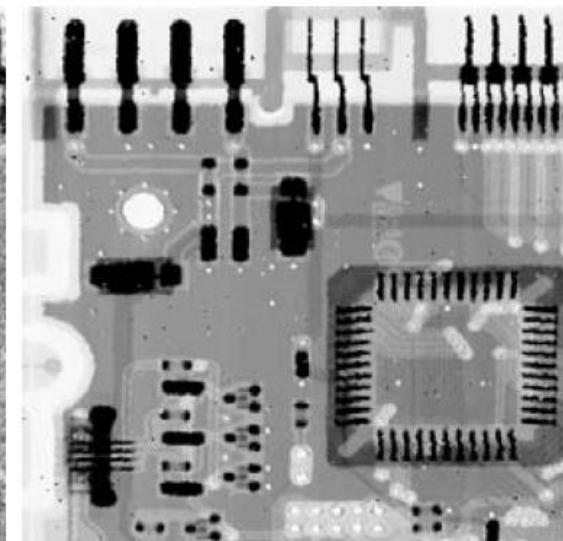
original:



average-filtered (3×3):



median-filtered (3×3):

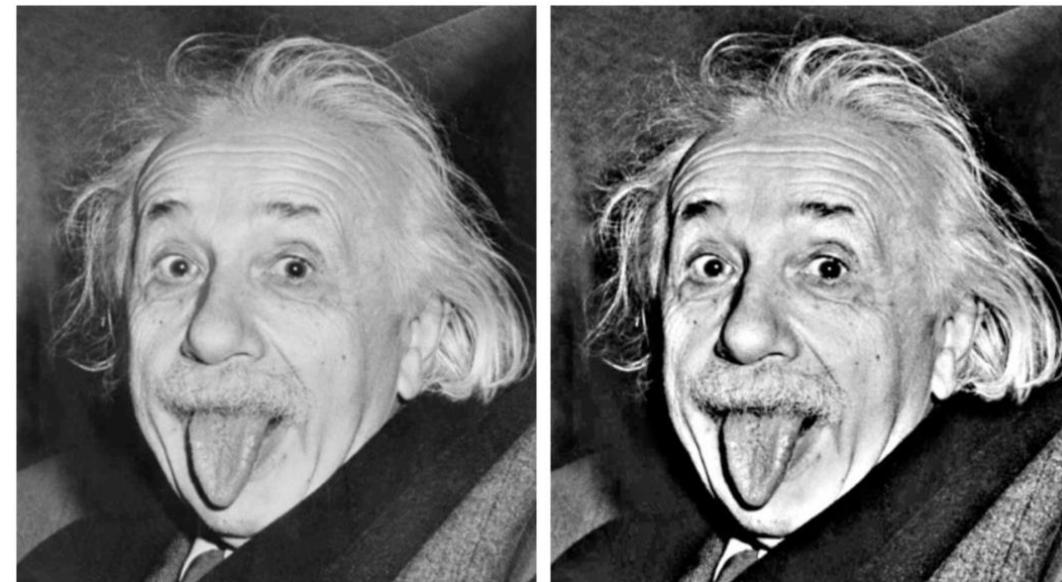


Sharpening Filters

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:

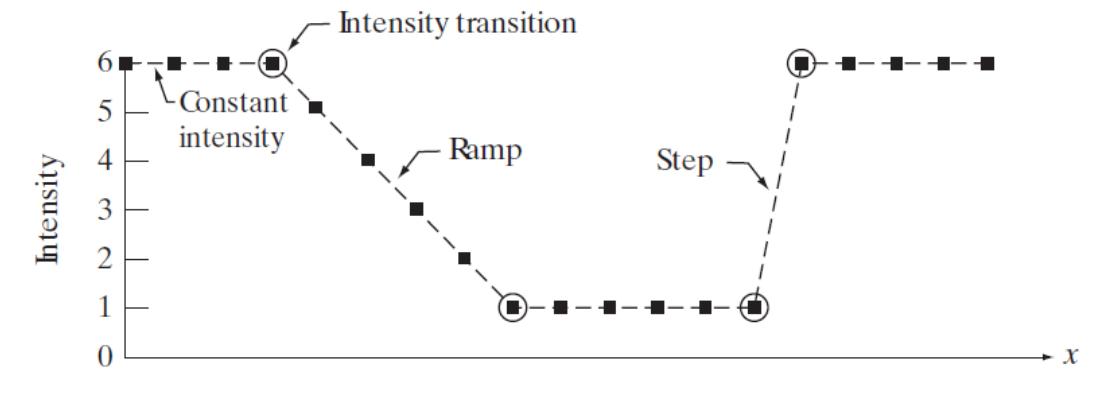
$$\frac{\partial f}{\partial x} = f(x + 1, y) - f(x, y)$$

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

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

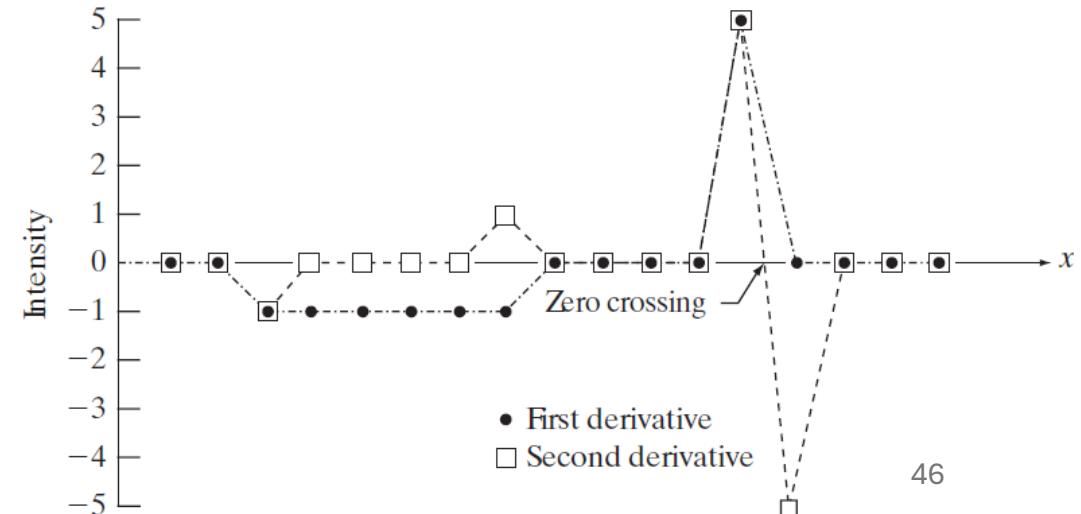
$$\frac{\partial^2 f}{\partial y^2} = f(x, y + 1) + f(x, y - 1) - 2f(x, y)$$

1D visualization:



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Scan line	6	6	6	6	5	4	3	2	1	1	1	1	1	1	6	6	6
1st derivative	0	0	-1	-1	-1	-1	-1	0	0	0	0	0	0	5	0	0	0

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
2nd derivative	0	0	-1	0	0	0	0	1	0	0	0	0	0	5	-5	0	0



Approximated Image Gradient

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

→ 1D derivative filter:

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

Sobel filters:

smoothing

$$\begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 1 \\ \hline\end{array}$$

horizontal

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

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

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

(convolution kernels)

construct

separable filters:

vertical

$$\begin{array}{|c|}\hline -1 \\ \hline 0 \\ \hline 1 \\ \hline\end{array}$$

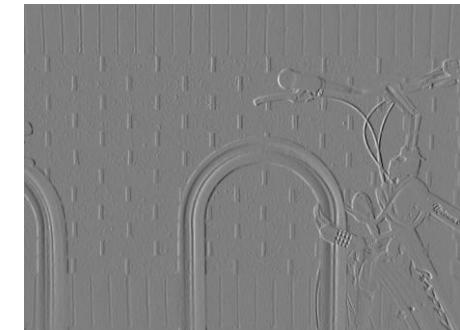
smoothing

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

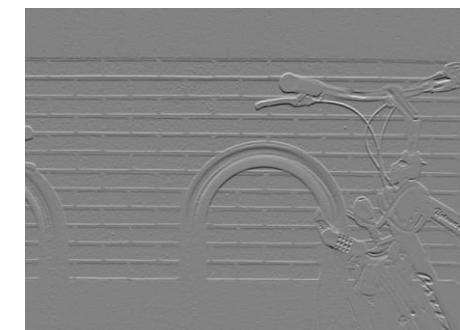
$$\cdot \frac{1}{8}$$

→ gradient with smoothing

convolved with an image:



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



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

Magnitude of Image Gradient

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

for sharpening:
add this to original



Different Gradient Filters

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

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

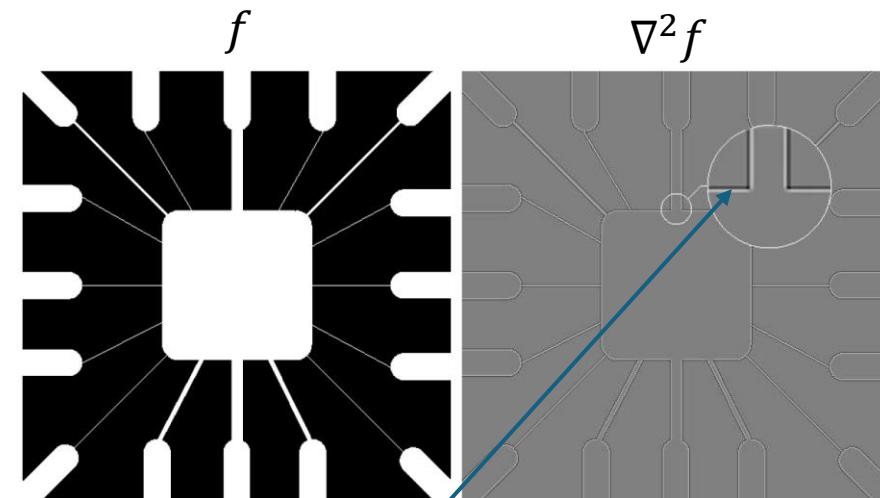
$$= f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

isotropic for rotations in increments of 90°

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

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

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



scaled to visualize negative values

zero-crossings \rightarrow edges

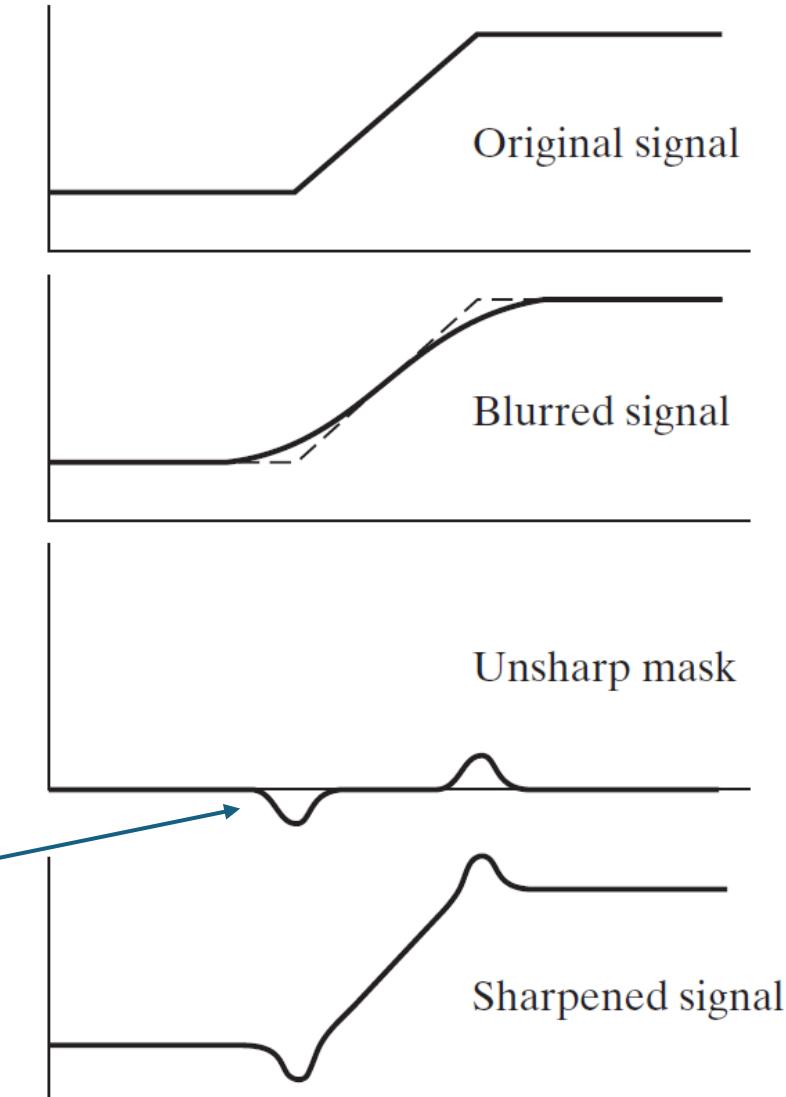
1	1	1
1	-8	1
1	1	1

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





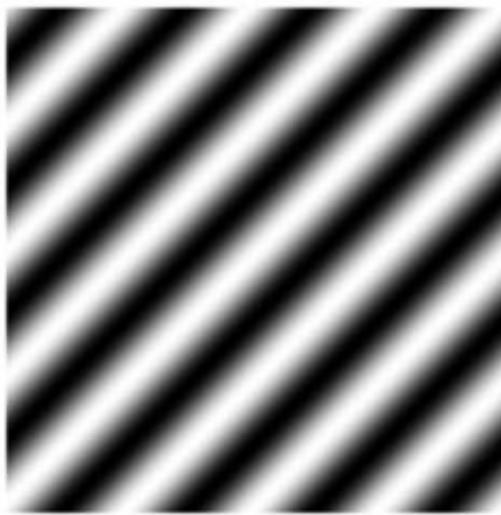
Fourier Transform

Fourier analysis allows to determine frequency characteristics of images or filters

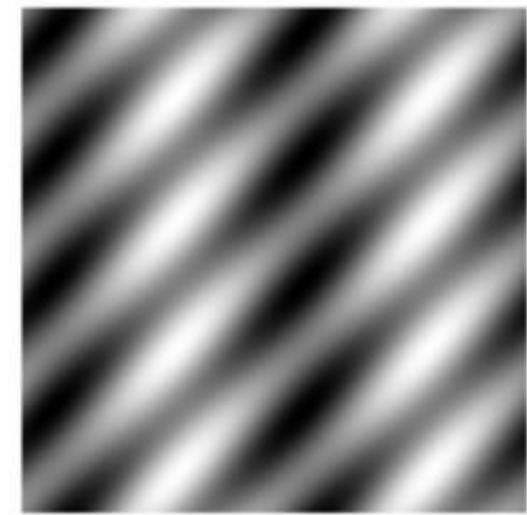
image



+



=



frequency
content



Frequency Representation of Spatial Function

response of a spatial function (filter) $h(x)$ to a sinusoid with frequency ω :

angular frequency $\omega = 2\pi f$
notation: f sometimes u or μ

$$h(x) * e^{i\omega x} = Ae^{i(\omega x + \phi)}$$

Fourier transform: tabulation of magnitude A and phase ϕ response **at each frequency**

$$H(\omega) = \mathcal{F}\{h(x)\} = Ae^{i\phi}$$

complex: needs to capture both amplitude and phase

Fourier transform pair: $h(x) \xleftrightarrow{\mathcal{F}} H(\omega)$

frequency domain

Fourier Transform Formulas

for continuous signals:

$$H(\omega) = \int_{-\infty}^{\infty} h(x)e^{-i\omega x}dx$$

inverse Fourier transforms:

$$h(x) = \int_{-\infty}^{\infty} H(\omega)e^{i\omega x}d\omega$$

Discrete Fourier Transform (DFT):

$$H(k) = \sum_{x=0}^{N-1} h(x)e^{-i\frac{2\pi kx}{N}}$$

unique only in range $k \in \left[-\frac{N}{2}, \frac{N}{2}\right]$

$$h(x) = \frac{1}{N} \sum_{k=0}^{N-1} H(k)e^{i\frac{2\pi kx}{N}}$$

2D Fourier Transforms

$$H(\omega_x, \omega_y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(x, y) e^{-i(\omega_x x + \omega_y y)} dx dy$$

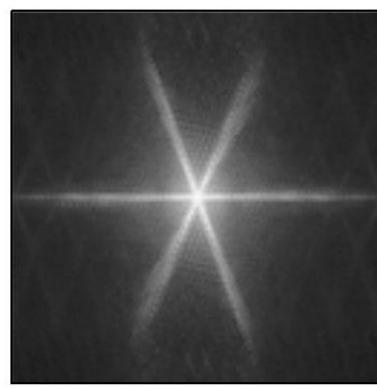
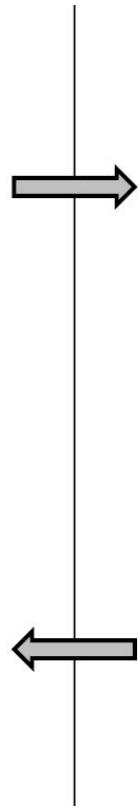
$$H(k_x, k_y) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} h(x, y) e^{-i2\pi(k_x x/M + k_y y/N)}$$



e.g., digital images (M and N as width and height in pixels)

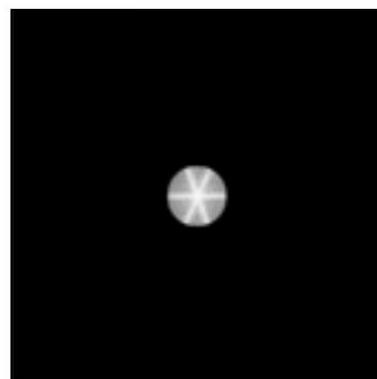
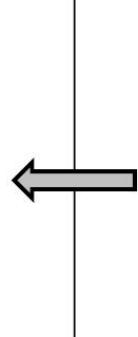
Filtering in Frequency Domain

low-pass filtering:

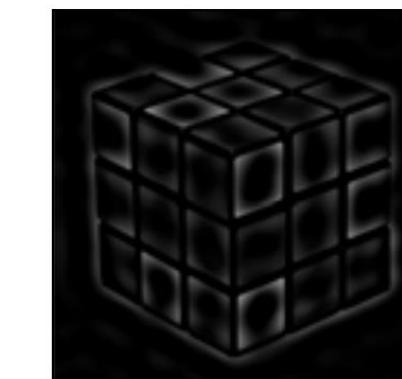


magnitude

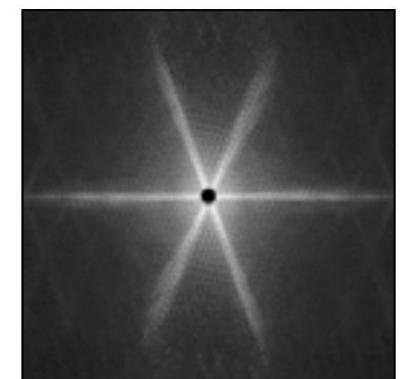
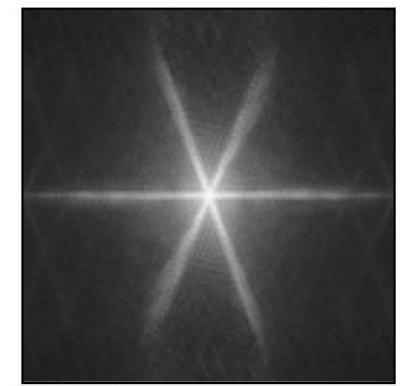
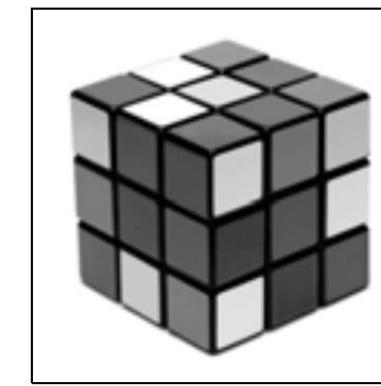
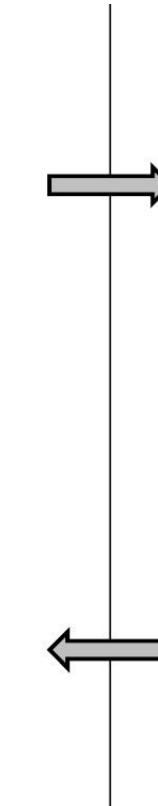
high-pass filtering:



→ smoothing

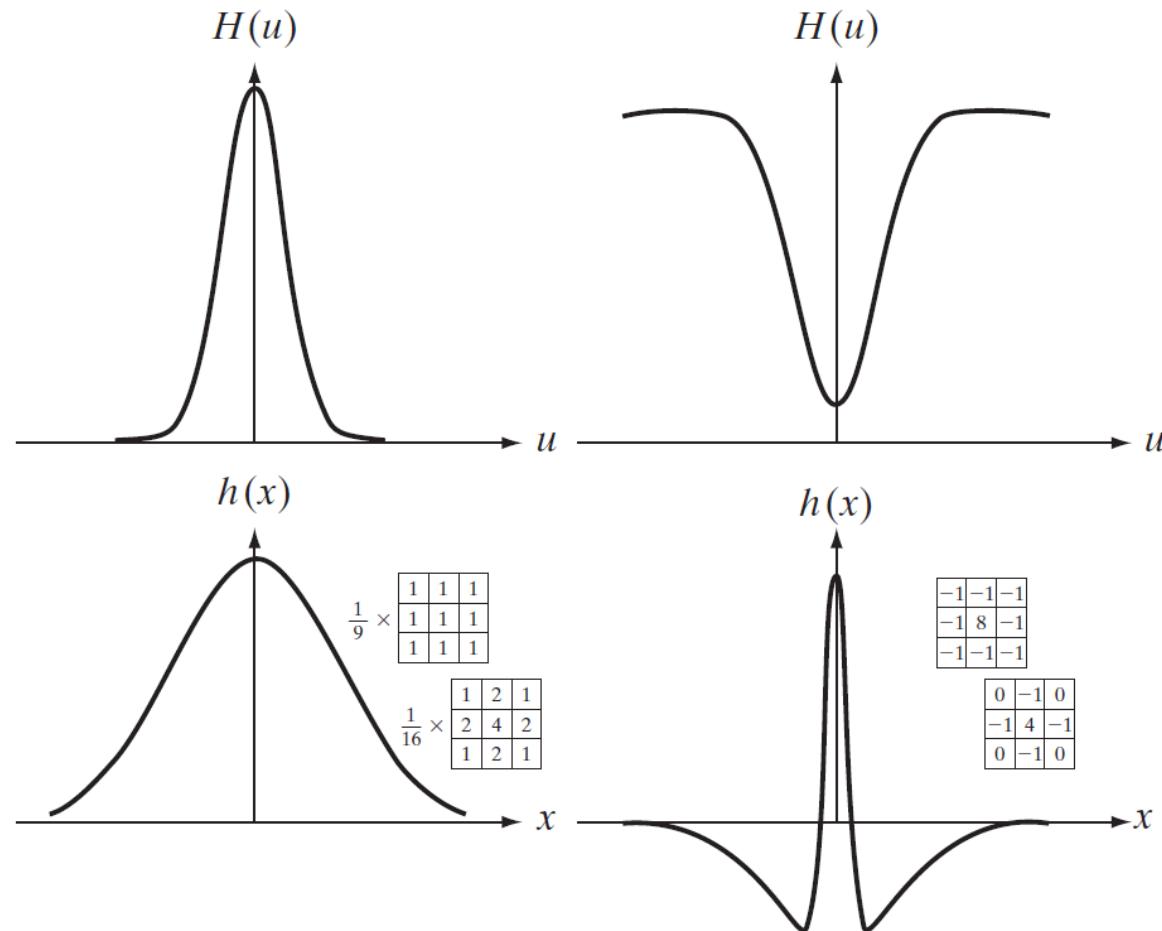


→ sharpening (if added to original)



Gaussian Filters in Frequency Domain

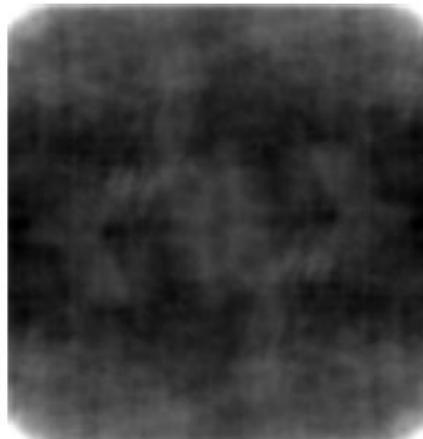
low-pass filter



Importance of Phase



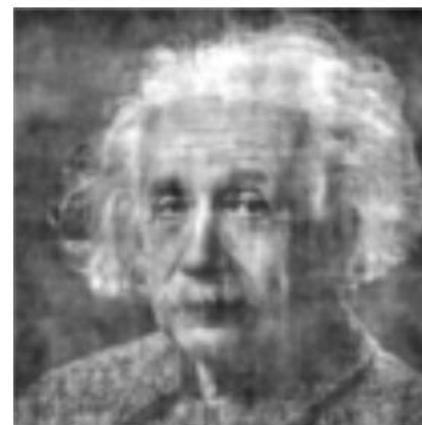
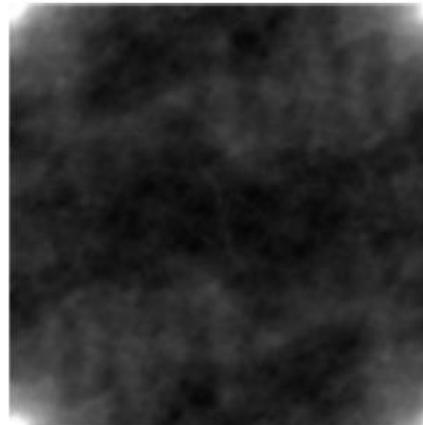
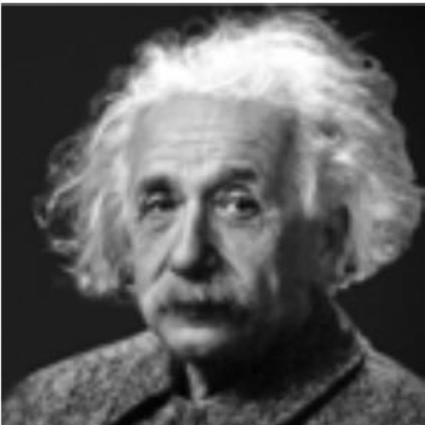
Original Image



Magnitude Preserved,
Phase Set to Zero



Phase Preserved,
Magnitude Set to Average
of Natural Images



Convolution Theorem

spatial domain:

$$g(x) = f(x) * h(x) \quad \leftrightarrow \quad G(\omega) = F(\omega) \cdot H(\omega)$$

$$g(x) = f(x) \cdot h(x) \quad \leftrightarrow \quad G(\omega) = F(\omega) * H(\omega)$$

frequency domain:

Convolution via Fourier Transform

$$g(x) = f(x) * h(x)$$

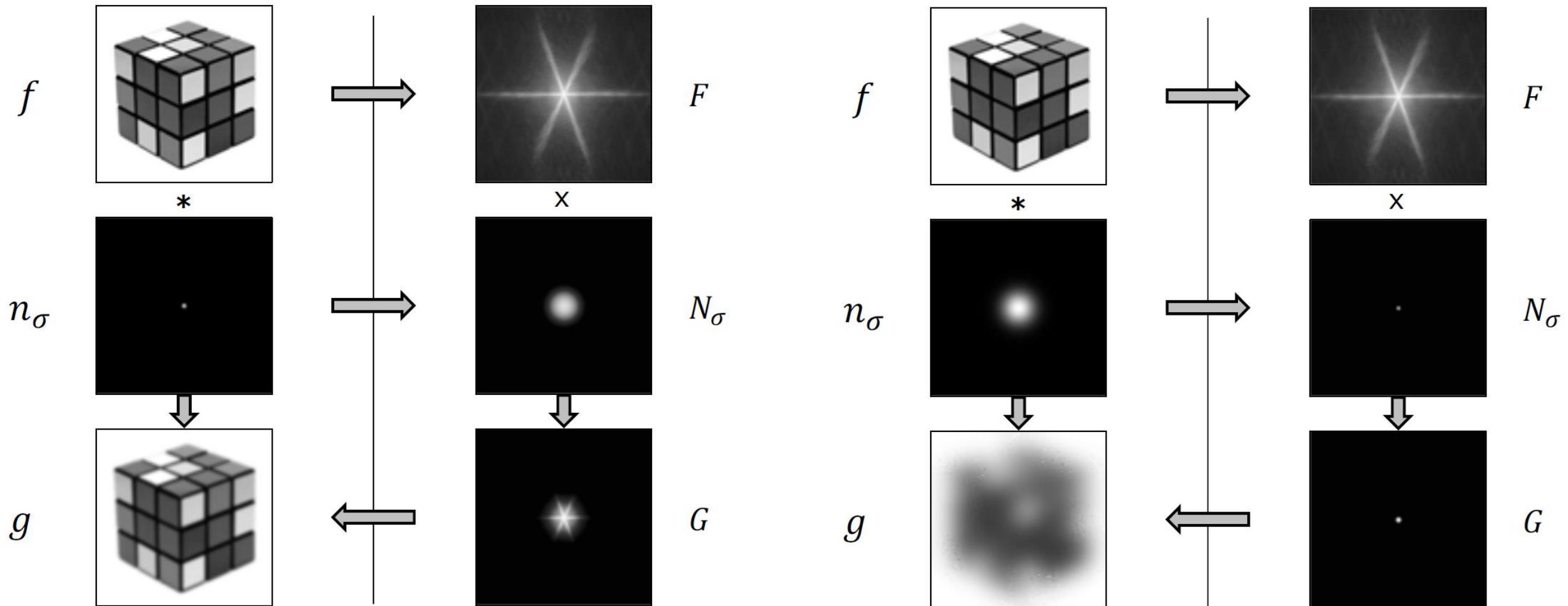
$$\mathcal{F}^{-1} \uparrow \quad \downarrow \mathcal{F} \quad \downarrow \mathcal{F}$$

$$G(\omega) = F(\omega) \cdot H(\omega)$$

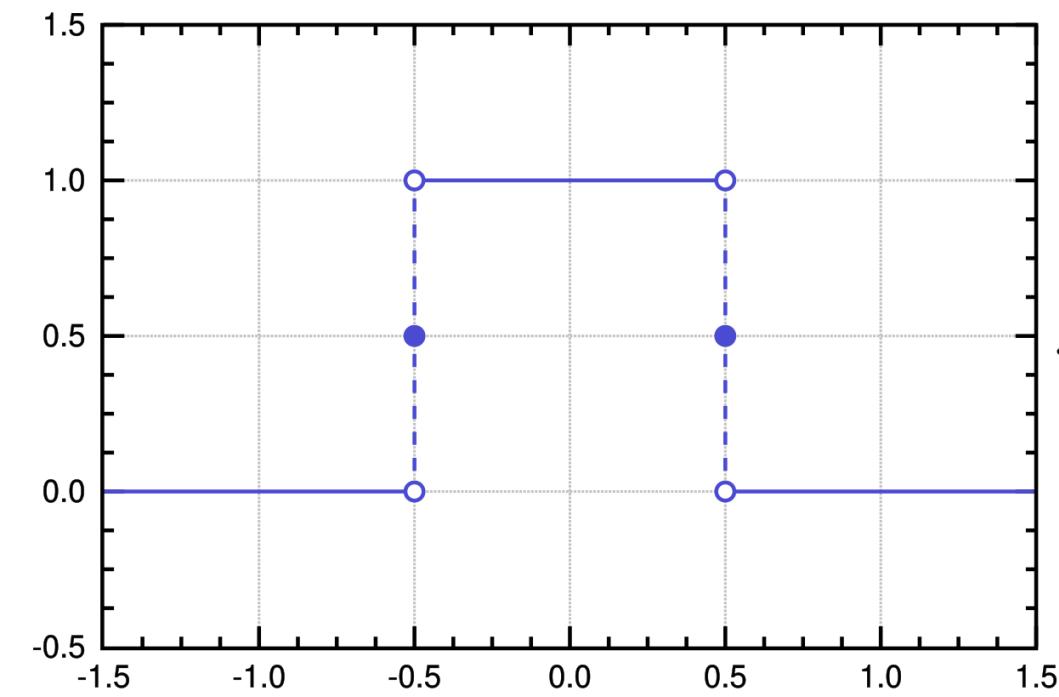
convolution of images with large spatial filters can be computationally heavy
→ more efficient in frequency domain, thanks to Fast Fourier Transform (enabling convolution times independent of filter size)

also, construction of filters often easier in frequency domain

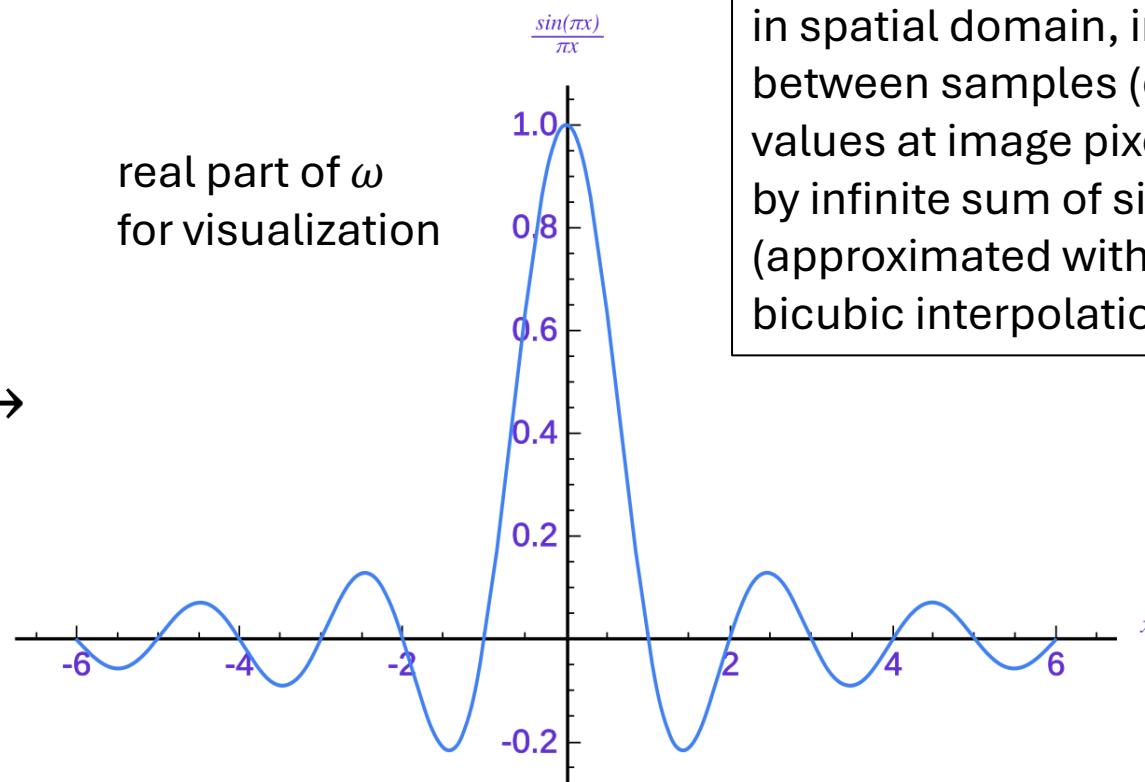
Example: Gaussian Smoothing



Fourier Transform Pair: Box and sinc Function



real part of ω
for visualization

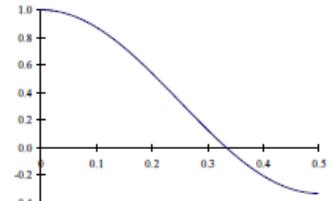
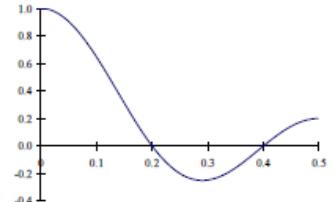
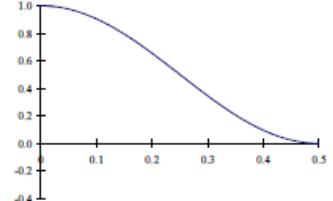
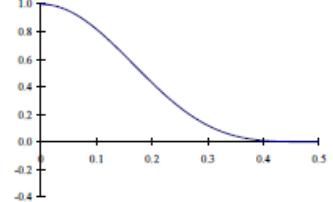
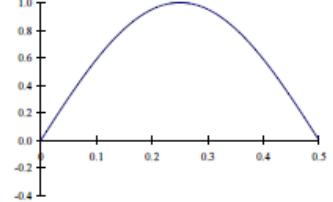
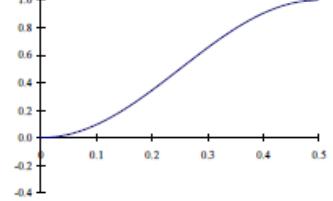


in spatial domain, interpolations
between samples (e.g., intensity
values at image pixels) formed
by infinite sum of sinc functions
(approximated with bilinear or
bicubic interpolations)

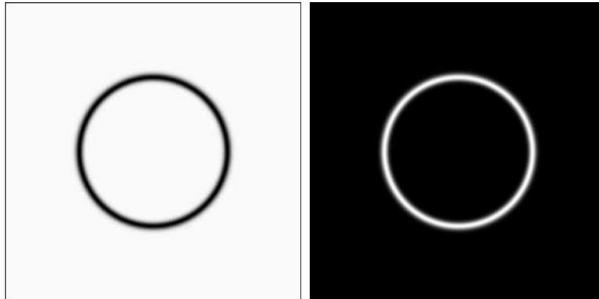
inverse scaling (valid for any function):
wider in spatial domain \rightarrow more compressed in frequency domain (and vice versa)
(e.g., narrow and broad Gaussian)

smoothing filters / low-pass filters

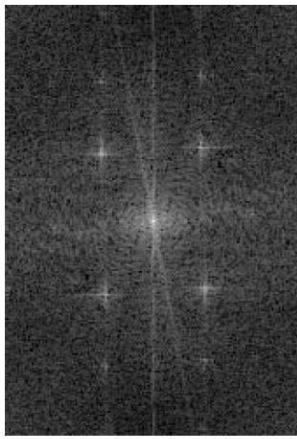
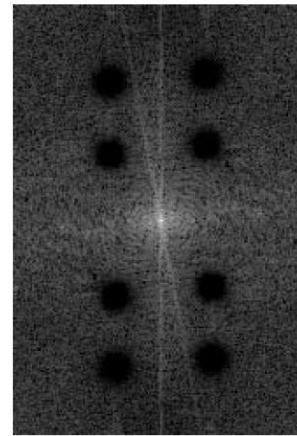
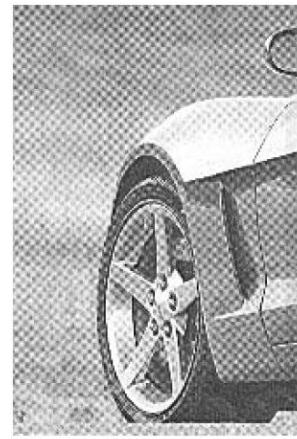
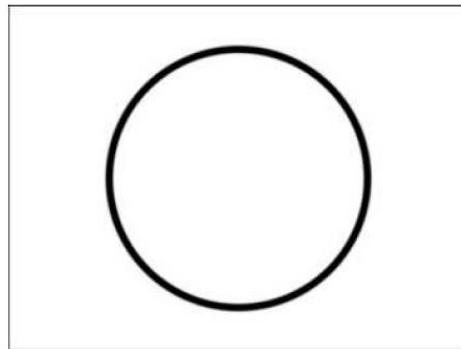
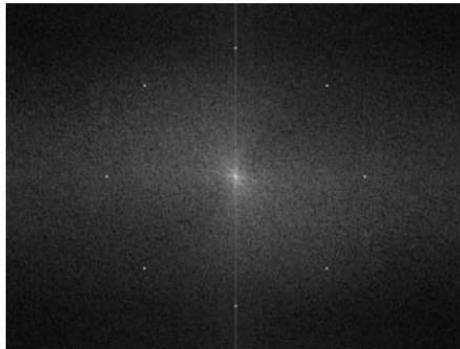
sharpening filters / high-pass filters

Name	Kernel	Transform	Plot
box-3	$\frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$	$\frac{1}{3}(1 + 2 \cos \omega)$	
box-5	$\frac{1}{5} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \end{bmatrix}$	$\frac{1}{5}(1 + 2 \cos \omega + 2 \cos 2\omega)$	
linear	$\frac{1}{4} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$	$\frac{1}{2}(1 + \cos \omega)$	
binomial	$\frac{1}{16} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \end{bmatrix}$	$\frac{1}{4}(1 + \cos \omega)^2$	
Sobel	$\frac{1}{2} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$	$\sin \omega$	
corner	$\frac{1}{2} \begin{bmatrix} -1 & 2 & -1 \end{bmatrix}$	$\frac{1}{2}(1 - \cos \omega)$	

Band-Reject and Band-Pass Filters



periodic noise can be effectively removed with band-reject filters in frequency domain



notch reject filter

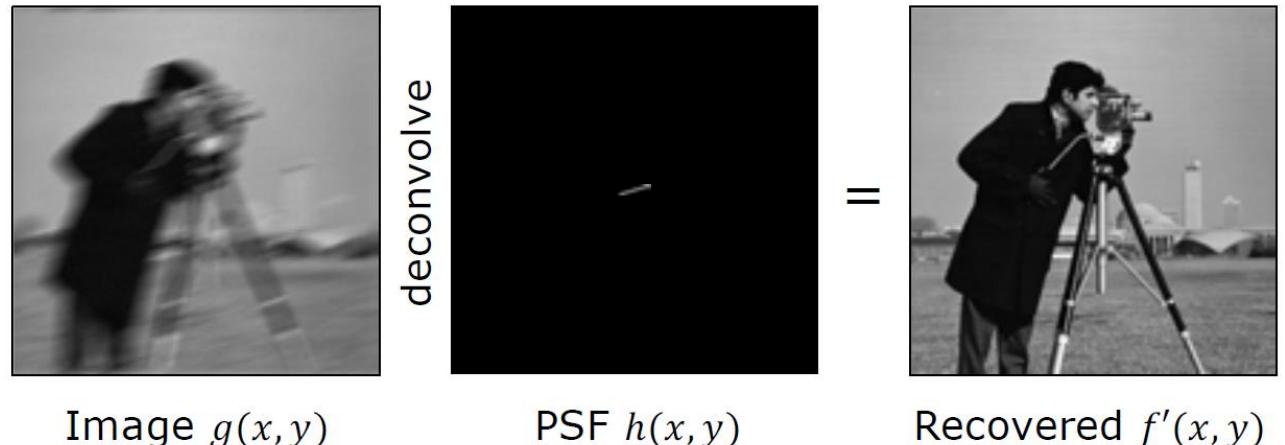
Deconvolution

application of convolution theorem:
inverse filtering in frequency domain
to get rid of degradation effect H
(image restoration)

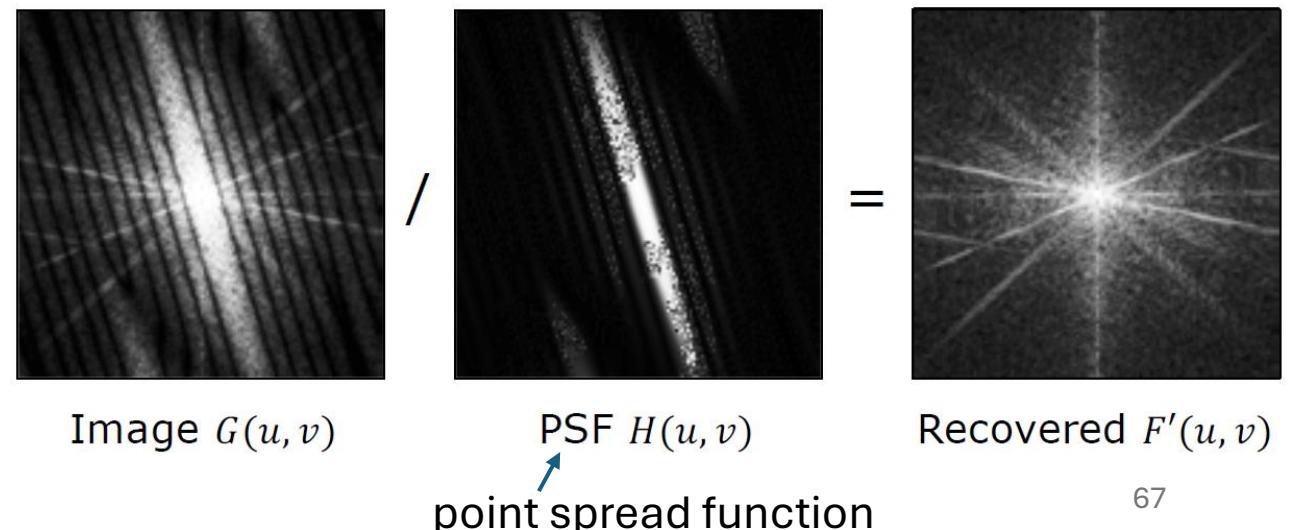
more complicated in practice:

- degradation function H usually not known completely → need for estimation
- usually also additive noise on top → need for noise suppression during deconvolution (Wiener filter)

example: motion blur

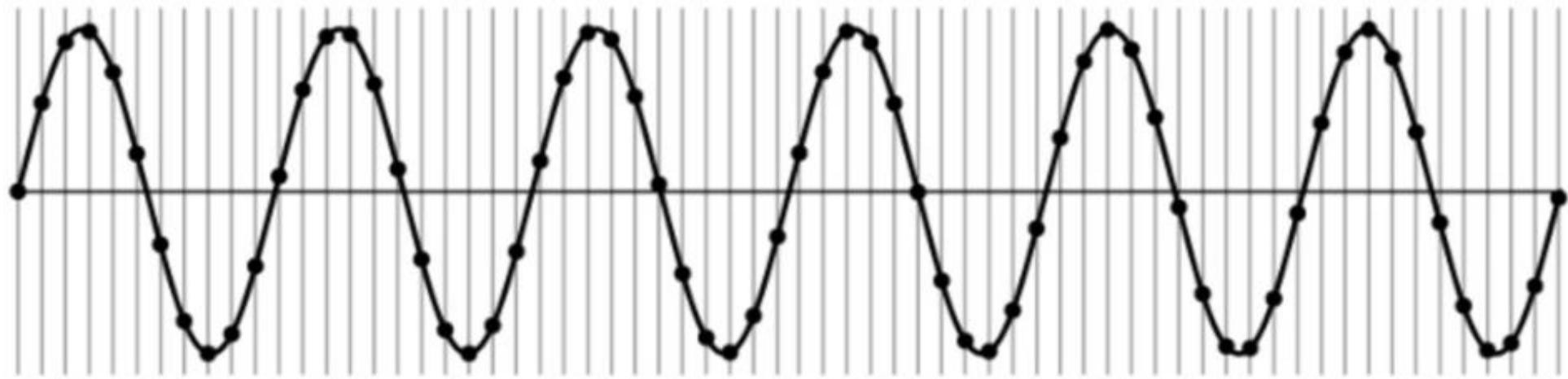


$$F'(u, v) = \frac{G(u, v)}{H(u, v)} \rightarrow \boxed{\text{IFT}} \rightarrow f'(x, y)$$



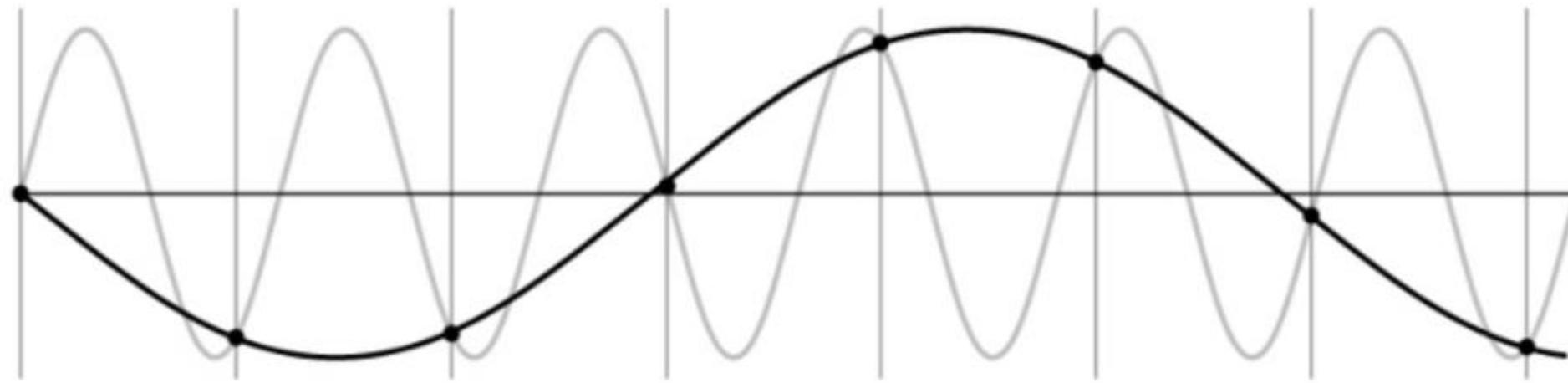
Aliasing

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

Sampling Theorem

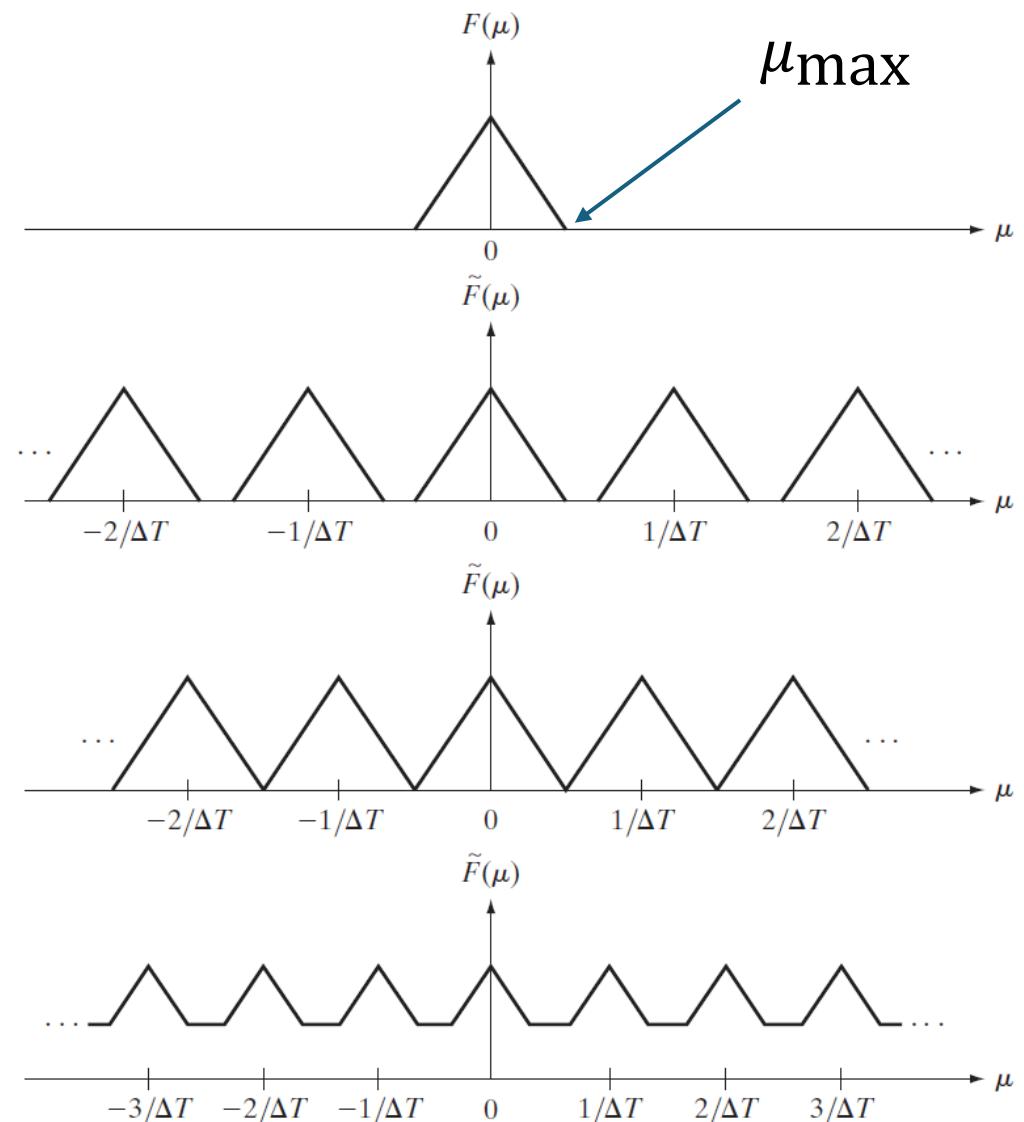
continuous signal can be recovered from its samples, if sampling rate exceeds Nyquist frequency:

$$\frac{1}{\Delta T} > 2\mu_{\max}$$

→ no aliasing

digital images: no aliasing if $\frac{N}{2} > \mu_{\max}$,
with number of pixels N
(second dimension accordingly)

Fourier transform of band-limited function:



Anti-Aliasing in Digital Images

no problem if signal is oversampled: number of pixels higher than twice the maximum frequency of image (for both width and height)

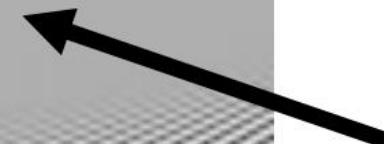
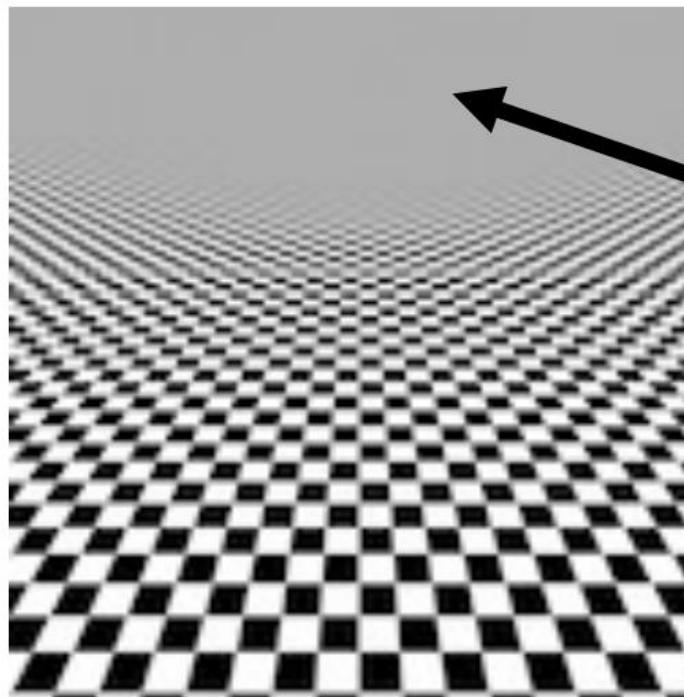
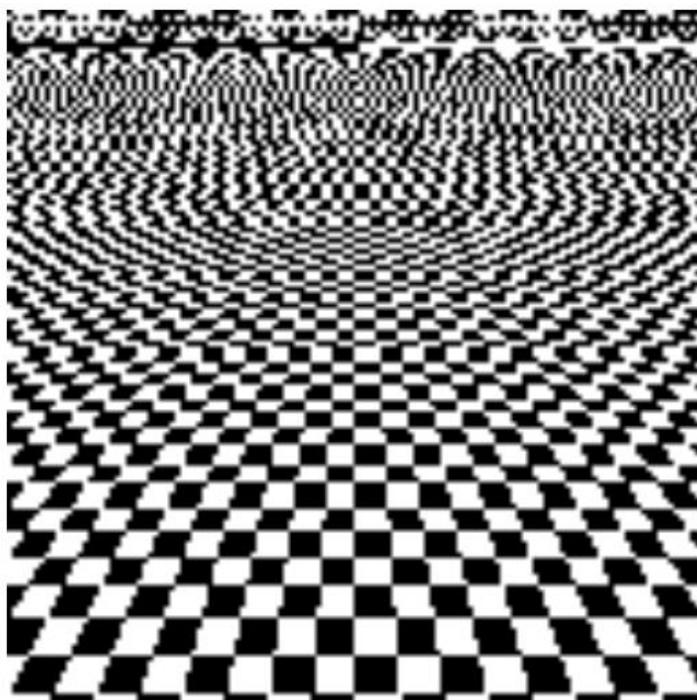
if not oversampled: smooth the signal

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

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

Example: Shrinking

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



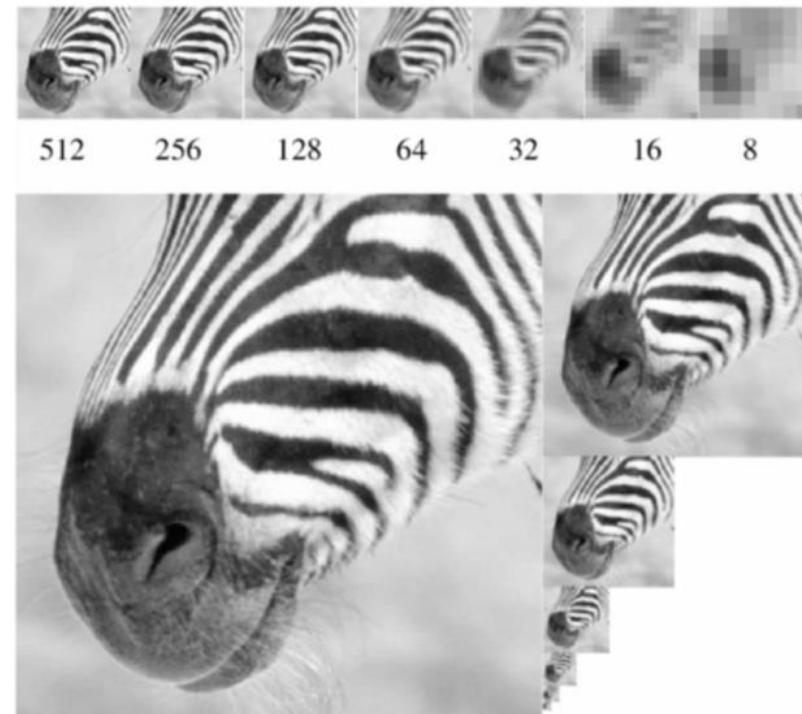
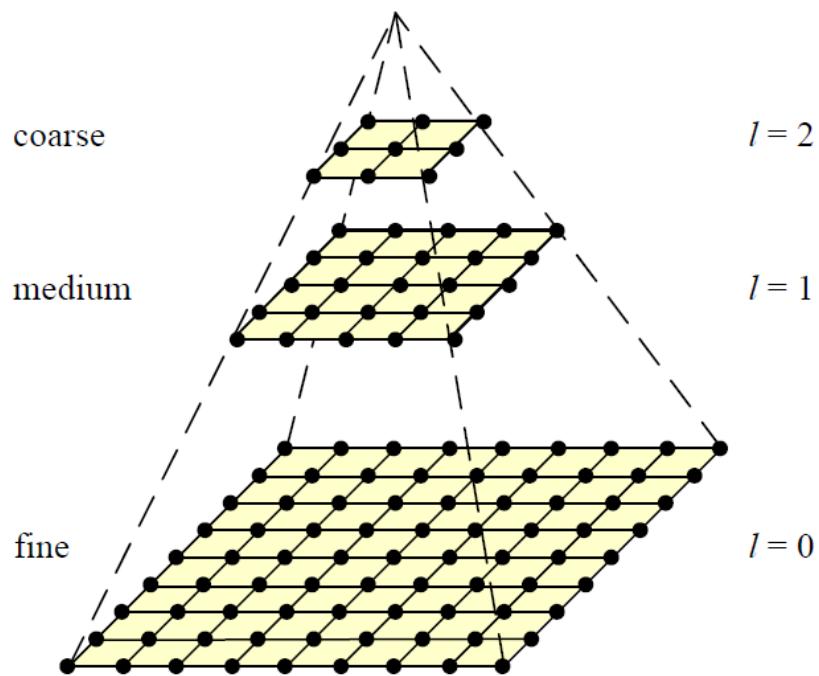
average of black
and white

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)

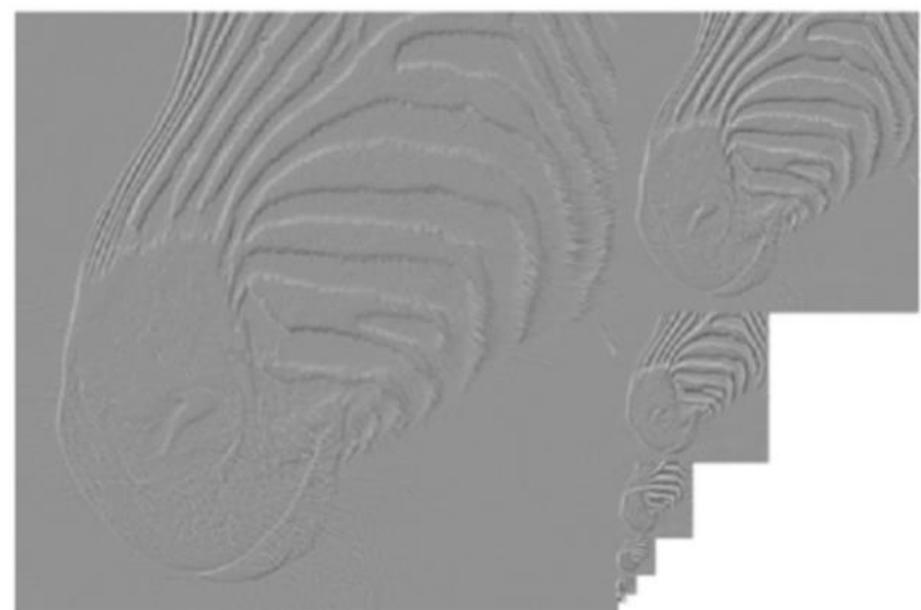


Laplacian Image Pyramid

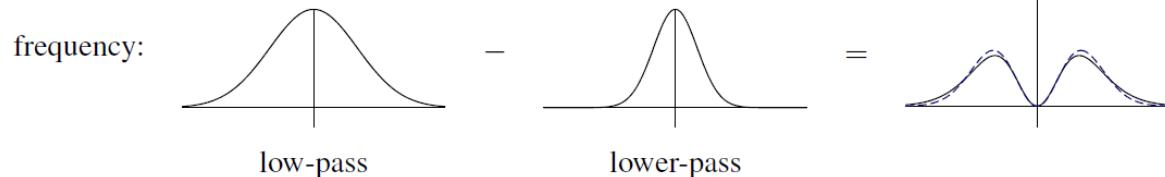
retain only residuals at different levels (allows later upsampling, starting from highest-level image)

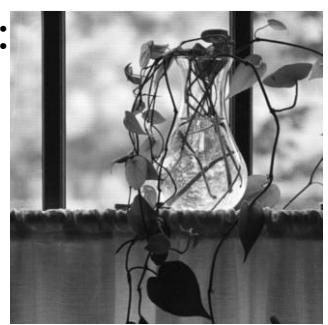


512 256 128 64 32 16 8



difference of Gaussians approximate Laplacian:





Wavelets

discrete wavelet transform:

- mapping of discrete function (e.g., a digital image) to sequence of coefficients (of a set of basis functions)
- performing sub-band coding (decomposition of an image into set of band-limited components) combined with progressive down-sampling (similar to residual pyramid)

four 128×128 sub-bands

smallest image approximation
(only low-pass filters)

four 64×64 sub-bands

four 256×256 sub-bands

- low-low (further split in next level)
- low-high (horizontal edges)
- high-low (vertical edges)
- high-high (mixed derivatives)

