Computer Vision

L02:Image Enhancement (Pre-processing Stage)

Prof. Dr. Mohammed Elmogy

Mansoura University
Faculty of Computers & Information
Dept. of Information Technology

elmogy@gmail.com

28. September 2025

Outline

Outline

Introduction

- The image processing is considered as the first stage in most computer vision applications, namely the use of image processing to preprocess the image and convert it into a form suitable for further analysis.

- Examples of image processing operations include exposure correction and color balancing, the reduction of image noise, increasing sharpness, or straightening the image by rotating it.

Some Common Image Processing Operations

(a) Original image; (b) Increased contrast

(a)

(b)

Some Common Image Processing Operations (cont.)

(c) Change in hue; (d) "Posterized" (quantized colors)

(c)

(d)

Some Common Image Processing Operations (cont.)

(e) Blurred; (f) Rotated

(e)

(f)

Image Noise

- Light Variations
- Camera Electronics
- Surface Reflectance
- Lens
- Time effects

Image Noise (cont'd)

- $I(x,y)$: the true pixel values
- $n(x,y)$: the noise at pixel (x,y)

$$\hat{I}(x,y) = I(x,y) + n(x,y)$$

Gaussian Noise

Image Noise (cont'd)

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

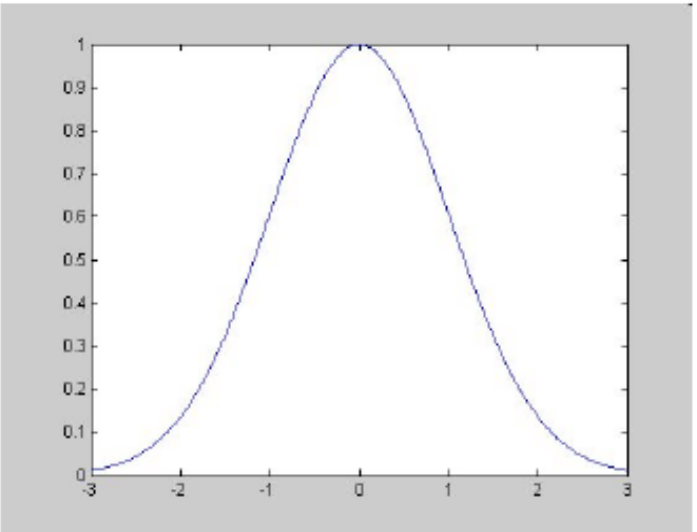


Image Processing Transformations (Filtering)

- Point Operators: point to point mapping to perform **brightness**, **contrast adjustments**,**color correction** and **transformations**.
- Image filters in spatial domain: mathematical operations of a grid of numbers to perform **smoothing**, **sharpening**, and **measuring texture**.
- Image filters in the frequency domain: a way to modify the frequencies of images to perform **denoising**, **sampling**, and **image compression**.
- Templates and Image Pyramids: a way to match a template to the image to perform **detection** and **coarse-to-fine registration**.

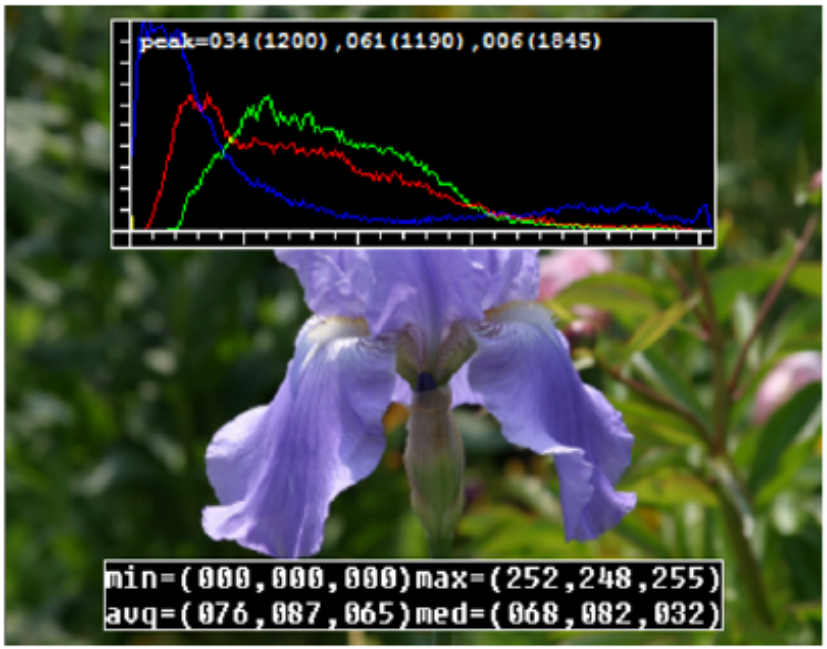
Outline

Point Operators/Processes

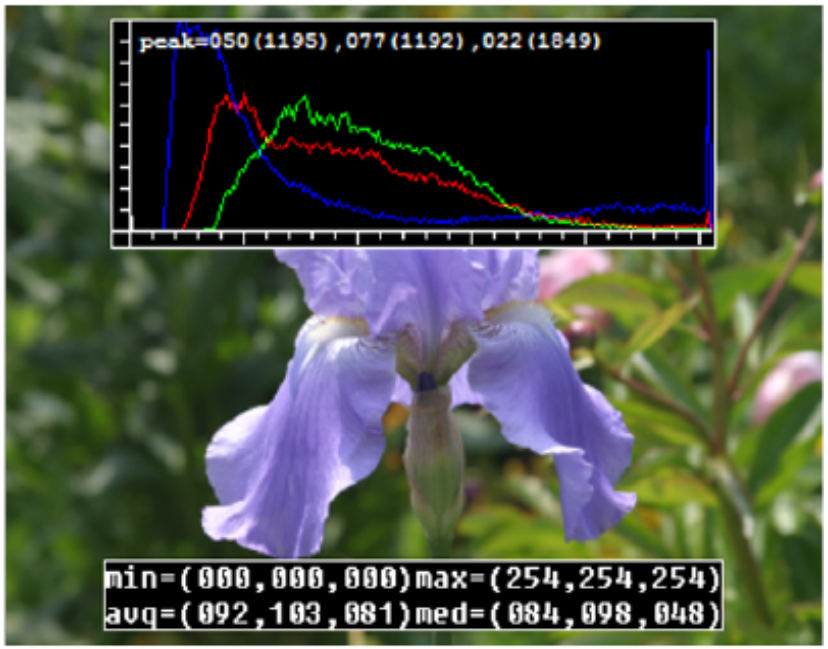
- They are the simplest kinds of image processing transforms, where each output pixel's value depends on only the corresponding input pixel value.
- Examples of such operators include brightness and contrast adjustments as well as color correction and transformations.

Some Local Image Processing Operations

(a) Original image along with its three color (per-channel) histograms; (b) Brightness increased (additive offset, b = 16)



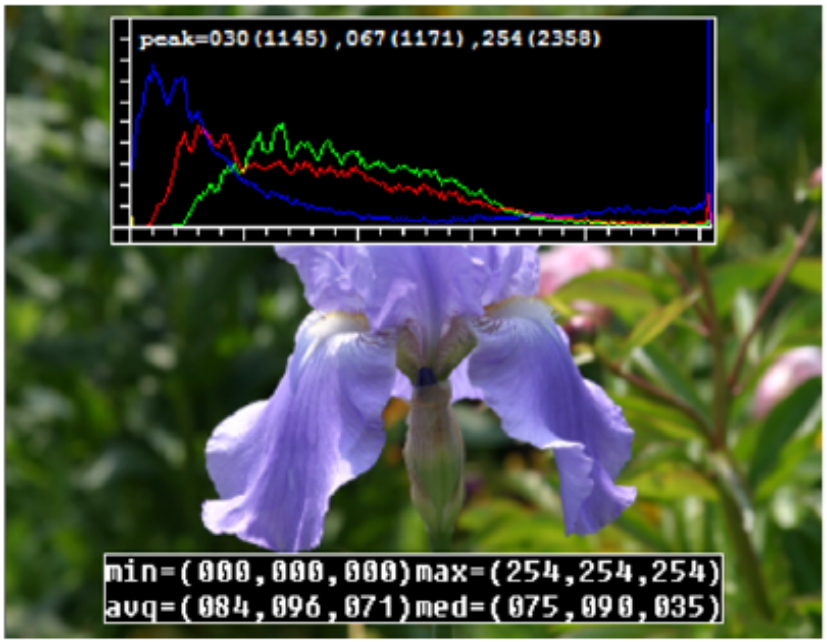
(a)



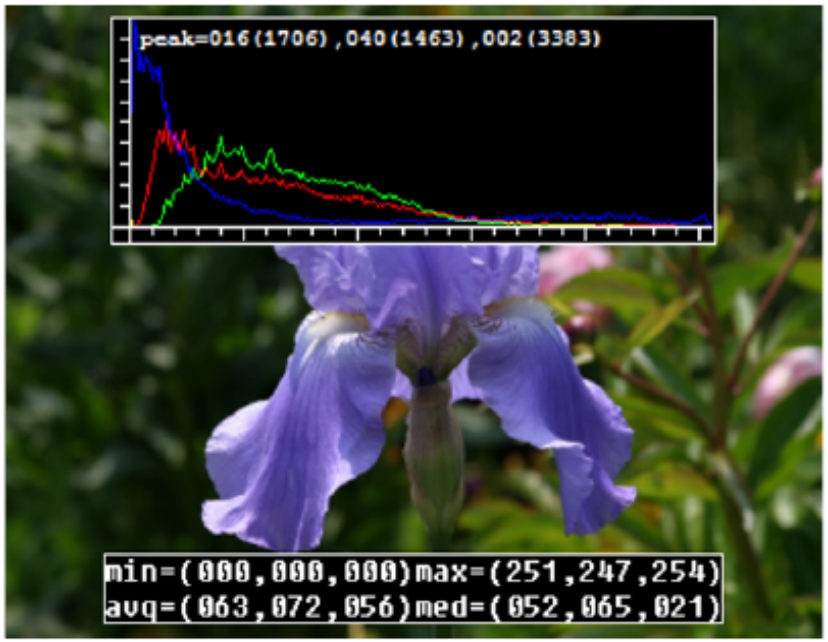
(b)

Some Local Image Processing Operations (cont.)

(c) Contrast increased (multiplicative gain, a = 1.1);(d) Gamma (partially) linearized ($\gamma = 1.2$)



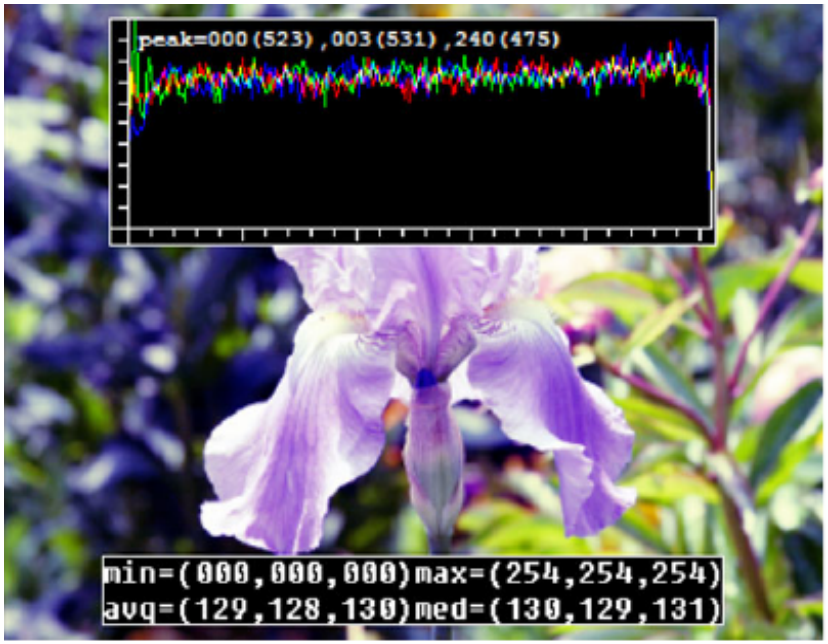
(c)



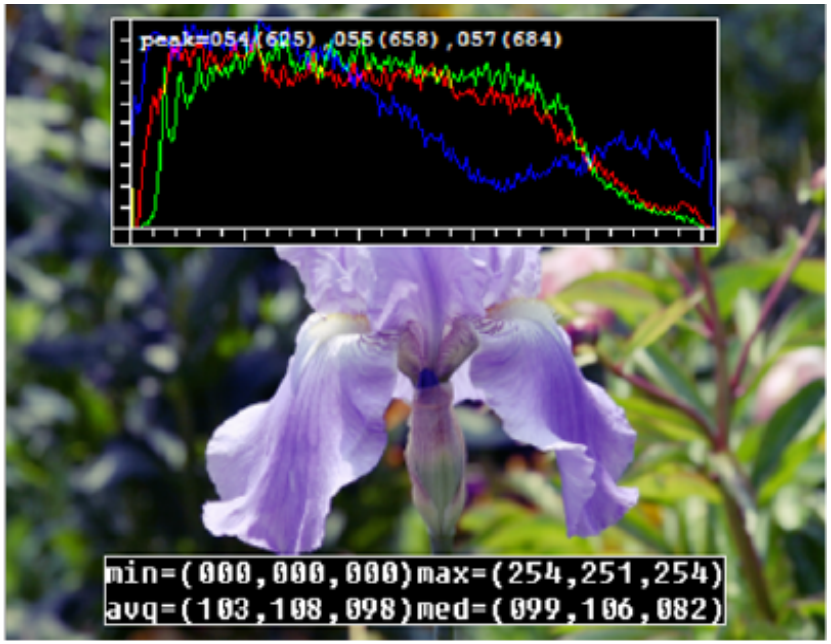
(d)

Some Local Image Processing Operations (cont.)

(e) Full histogram equalization; (f) Partial histogram equalization



(e)



(f)

Pixel Transforms

- A general image processing operator is a function that takes one or more input images and produces an output image. In the continuous domain, this can be denoted as: $g(x) = h(f(x))$ or $g(x) = h(f_0(x), ..., f_n(x))$.
- For discrete (sampled) images, the domain consists of a finite number of pixel locations, $x = (i, j)$, and we can write $g(i, j) = h(f(i, j))$.
- Two commonly used point processes are multiplication and addition with a constant, $g(x) = af(x) + b$. The parameters $a > 0$ and b are often called the gain and bias parameters; sometimes these parameters are said to control contrast and brightness.

Some Pixel Transforms

- The bias and gain parameters can also be spatially **varying**, $g(x) = a(x)f(x) + b(x)$, e.g., when simulating the graded density filter used by photographers to selectively darken the sky.
- The **linear blend operator**, $g(x) = (1 - \alpha)f_0(x) + \alpha f_1(x)$. By varying α from $0 \rightarrow 1$, this operator can be used to perform a temporal cross-dissolve between two images or videos.
- **Gamma correction** is used to remove the non-linear mapping between input radiance and quantized pixel values. To invert the gamma mapping applied by the sensor, we can use $g(x) = [f(x)]^{1/\gamma}$, where a gamma value of $\gamma \approx 2.2$ is a reasonable fit for most digital cameras.

Color Transforms

- While color images can be treated as arbitrary vector-valued functions or collections of independent bands, it usually makes sense to think about them as highly correlated signals with strong connections to the image formation process, sensor design, and human perception.
- In fact, adding the same value to each color channel not only increases the apparent intensity of each pixel, it can also affect the pixel's hue and saturation.
- Color balancing can be performed either by multiplying each channel with a different scale factor or by the more complex process of mapping to XYZ color space.

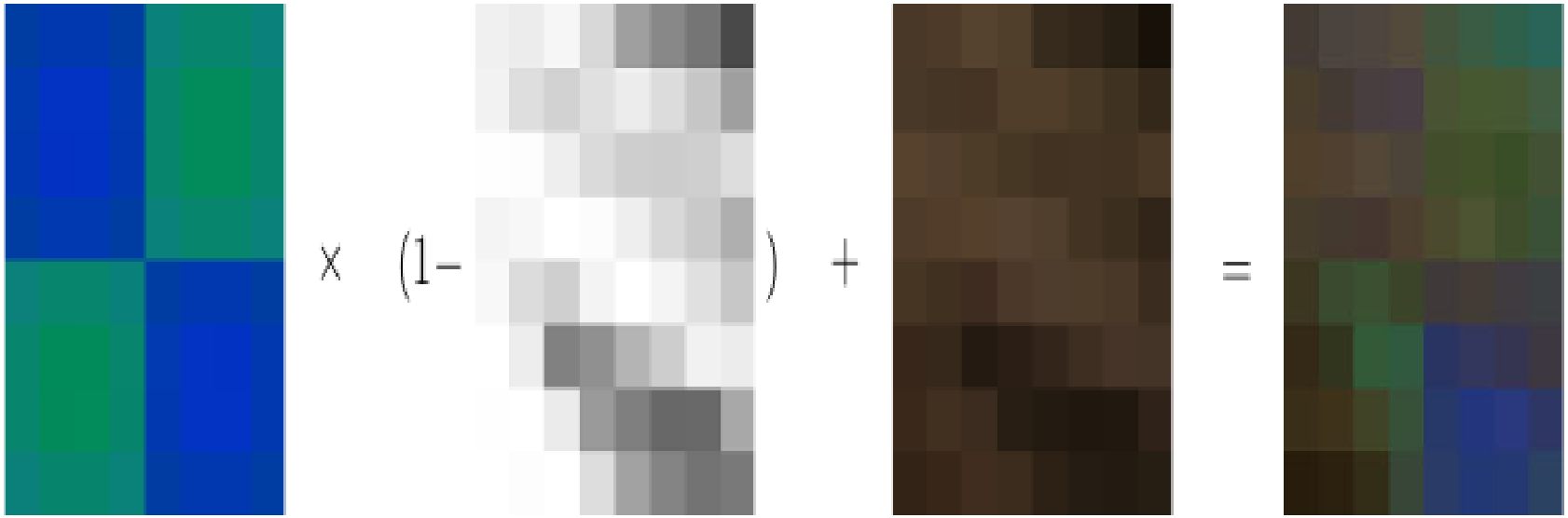
Compositing and Matting

- Matting: The process of extracting the object from the original image.
- Compositing: The process of inserting an image into another image.
- The intermediate representation used for the foreground object between these two stages is called an alpha-matted color image.

Image Matting and Compositing

(a) Source image; (b) Extracted foreground object F ; (c) Alpha matte α shown in grayscale; (d) New composite C

Compositing equation $C = (1 - \alpha)B + \alpha F$.



B α αF C

(a) (b) (c) (d)

Alpha-matted Image

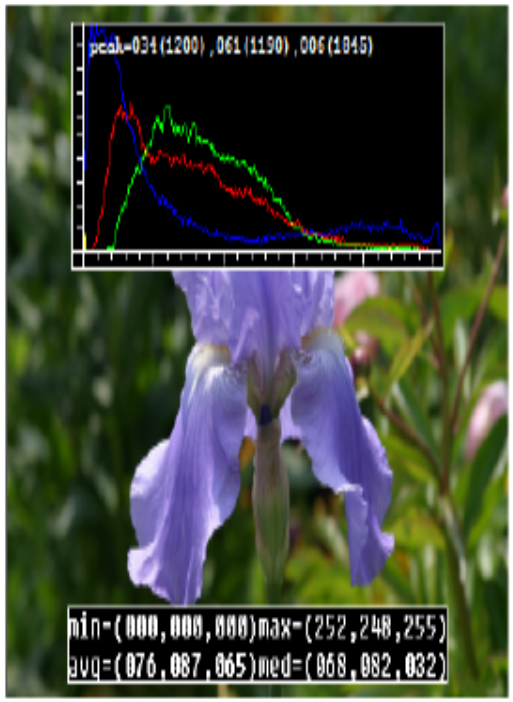
- It contains a fourth alpha channel α (or A) that describes the relative amount of opacity or fractional coverage at each pixel. The opacity is the opposite of the transparency.
- Pixels within the object are fully opaque $(\alpha = 1)$,while pixels fully outside the object are transparent $(\alpha = 0)$.
- To composite a new (or foreground) image on top of an old (background) image, the over operator is used, $C = (1 - \alpha)B + \alpha F$.
- This operator attenuates the influence of the background image B by a factor $(1 - \alpha)$ and then adds in the color (and opacity) values corresponding to the foreground layer F.

Histogram Equalization

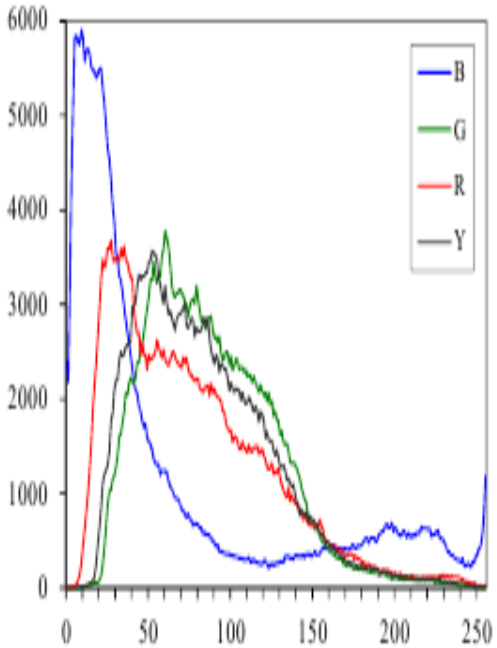
- Plot the histogram of the individual color channels and luminance values. We can compute relevant statistics, such as the minimum, maximum, and average intensity values.
- Histogram equalization: to find an intensity mapping function $f(I)$ such that the resulting histogram is flat.
- The trick to finding such a mapping is the same one that people use to generate random samples from a probability density function, which is to first compute the cumulative distribution function.

Histogram Analysis and Equalization

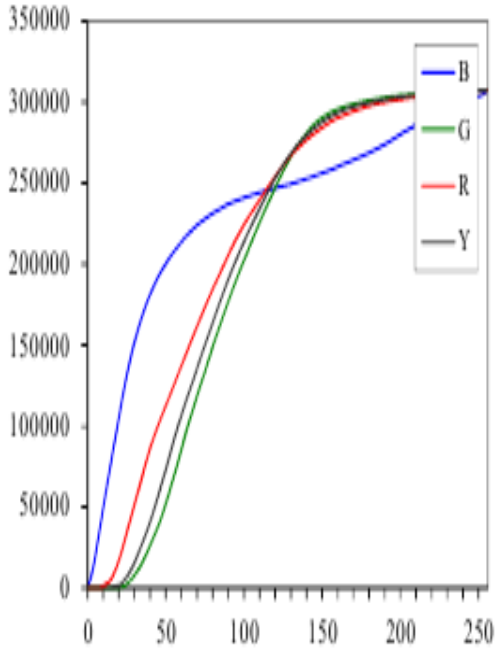
(a) Original image (b) Color channel and intensity (luminance) histograms; (c) Cumulative distribution functions



(a)



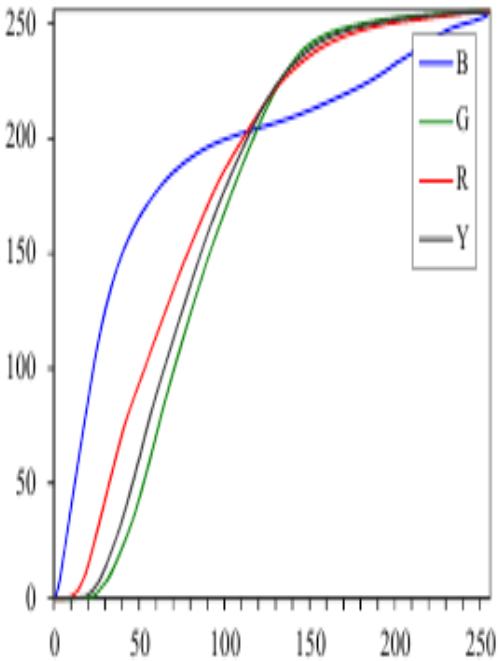
(b)



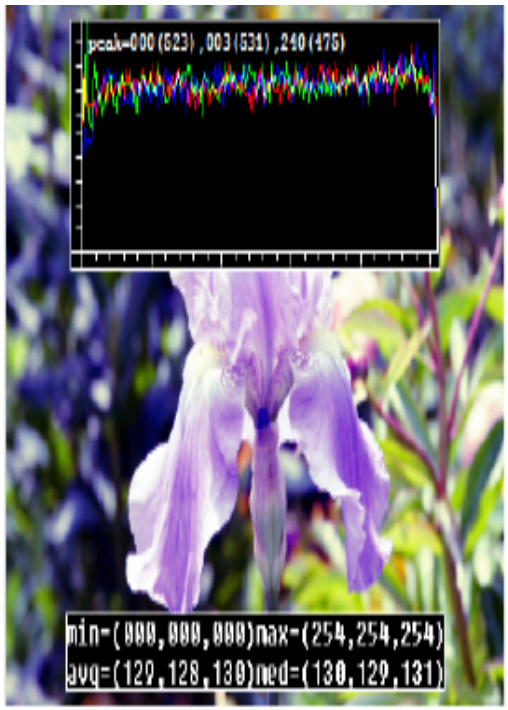
(c)

Histogram Analysis and Equalization (cont.)

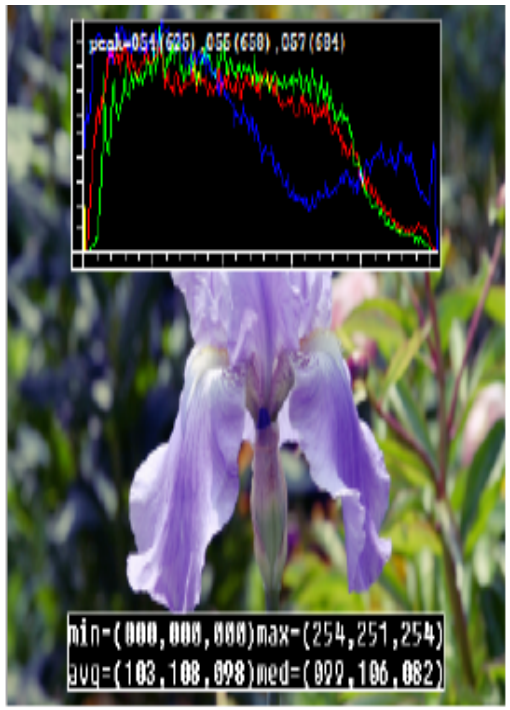
(d) Equalization (transfer) functions; (e) Full histogram equalization; (f) Partial histogram equalization



(d)



(e)



(f)

Outline

Linear Filtering

- Locally adaptive histogram equalization is an example of a neighborhood operator or local operator, which uses a collection of pixel values in the vicinity of a given pixel to determine its final output value.
- In addition to performing local tone adjustment, neighborhood operators can be used to filter images in order to add soft blur, sharpen details, accentuate edges, or remove noise.
- Linear filtering operators involve weighted combinations of pixels in small neighborhoods.

Neighborhood Filtering (Convolution)

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

$f(x,y)$

*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

$h(x,y)$

=

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

$g(x,y)$

Some Neighborhood Operations

(a) Original image; (b) Blurred

(a)

(b)

Some Neighborhood Operations (cont.)

(c) Sharpened; (d) Smoothed with edge-preserving filter

(c)

(d)

Image Filtering

- Image filtering: compute function of local neighborhood at each position.

- It can be used in:
  - **Enhance images**: Denoise, resize, increase contrast, etc.
  - **Extract information from images**: Texture, edges, distinctive points, etc.
  - **Detect patterns**: Template matching.

Definitions

- Derivative: **rate of change**. For example, **speed is a rate of change of a distance** and **acceleration is a rate of change of speed**.

- Average (Mean): dividing the sum of N values by N.

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x) = f_x$$

$$v = \frac{ds}{dt} \text{ speed} \qquad a = \frac{dv}{dt} \text{acceleration}$$

Discrete Derivative Finite Difference

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$ Backward difference

$$\frac{df}{dx} = f(x) - f(x+1) = f'(x)$$ Forward difference

$$\frac{df}{dx} = f(x+1) - f(x-1) = f'(x)$$ Central difference

Example

$f(x) =$	10	15	10	10	25	20	20	20
$f'(x) =$	0	5	−5	0	15	−5	0	0
$f''(x) =$	0	5	−10	5	15	20	5	0

Derivative Masks

Backward difference	[-1 1]
Forward difference	[1 -1]
Central difference	[-1 0 1]

Correlation

$$f \otimes h = \sum_k \sum_l f(k,l)h(i+k, j+l)$$

f = Image

h = Kernel

f

f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

$\otimes$

h

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

$$f * h = f_1h_1 + f_2h_2 + f_3h_3 + f_4h_4 + f_5h_5 + f_6h_6 + f_7h_7 + f_8h_8 + f_9h_9$$

Convolution

$$f * h = \sum_k \sum_l f(k,l)h(i-k, j-l)$$

f = Image

h = Kernel

h

h_1	h_2	h_3
h_4	h_5	h_6
h_7	h_8	h_9

$X - flip$

h_7	h_8	h_9
h_4	h_5	h_6
h_1	h_2	h_3

$Y - flip$

f

f_1	f_2	f_3
f_4	f_5	f_6
f_7	f_8	f_9

$\otimes$

h_9	h_8	h_7
h_6	h_5	h_4
h_3	h_2	h_1

$$f * h = f_1h_9 + f_2h_8 + f_3h_7 + f_4h_6 + f_5h_5 + f_6h_4 + f_7h_3 + f_8h_2 + f_9h_1$$

Box Filter

- Replaces each pixel with an average of its neighborhood.
- Achieve smoothing effect (remove sharp features).

$$g[\cdot,\cdot] = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Image Filtering

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

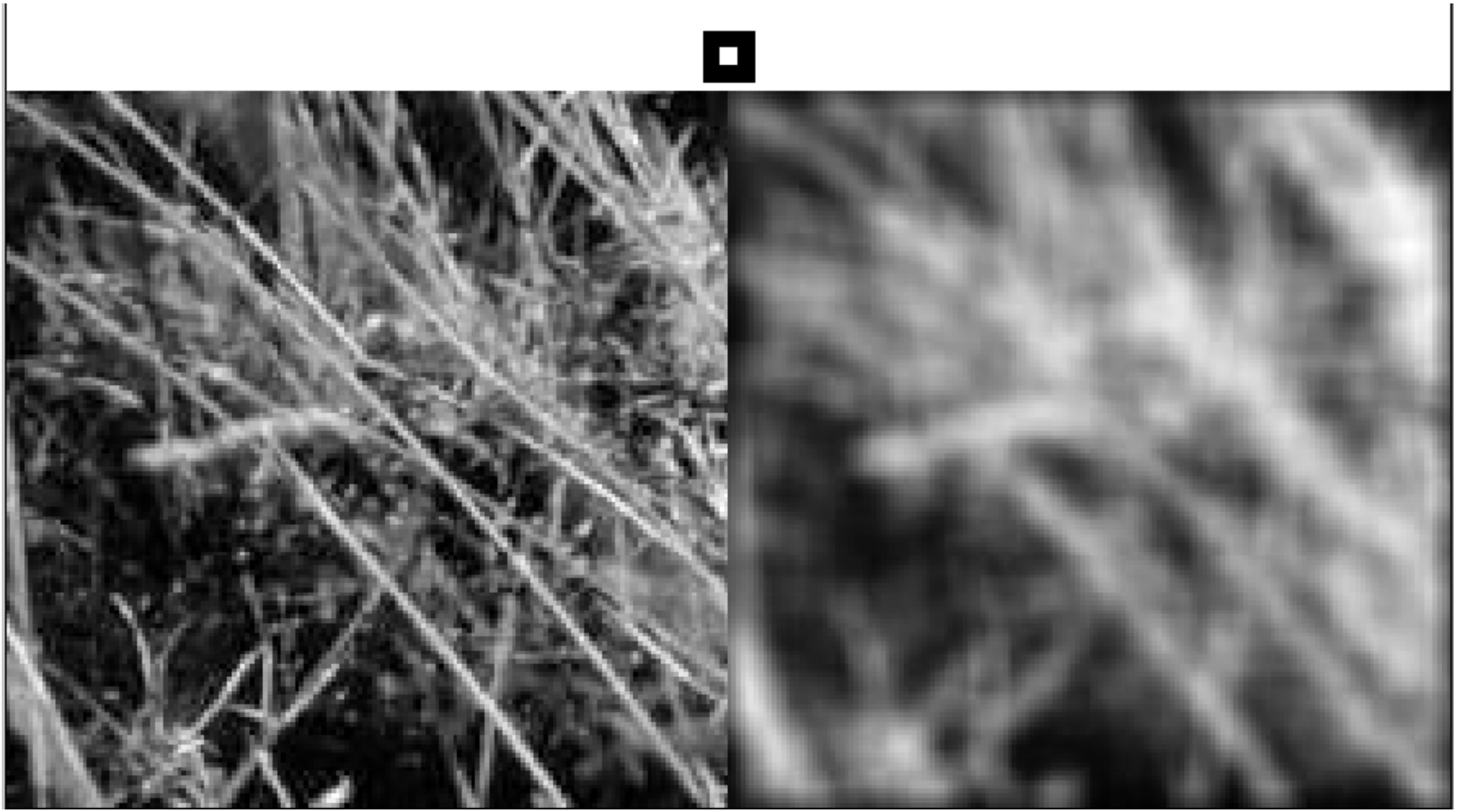
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$h[.,.]$

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

$$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$$

Smoothing with Box Filter



Practice with Linear Filters

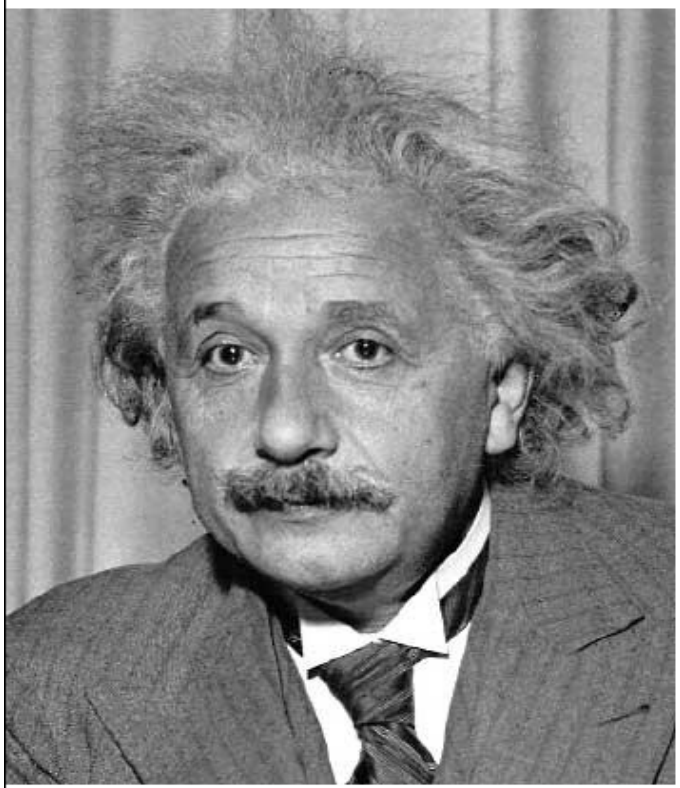
Original

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpening filter

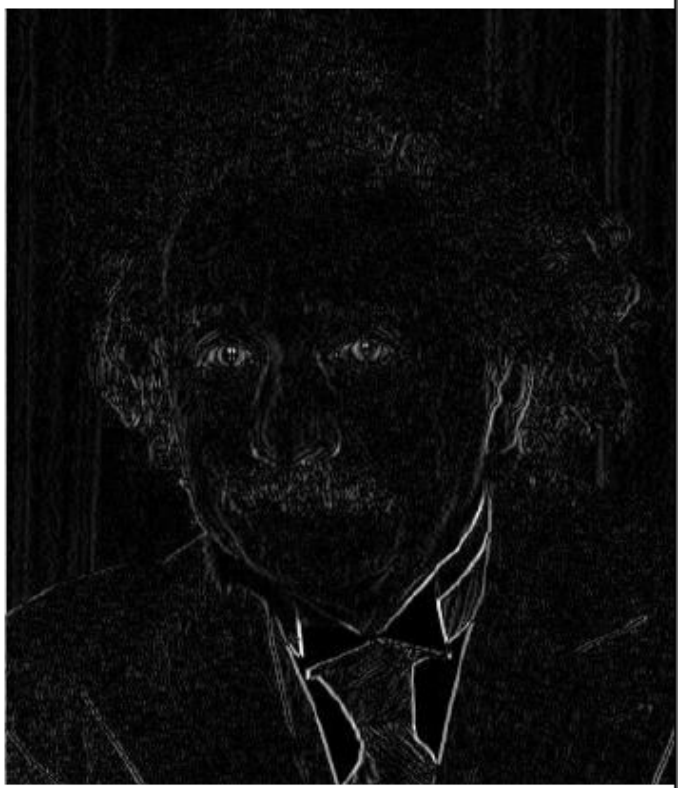
- Accentuates differences with local average

Other filters



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

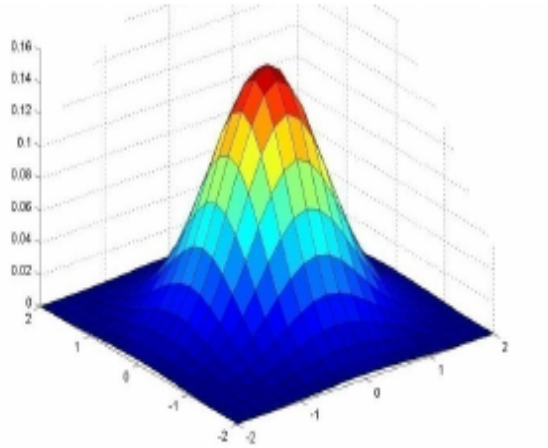
Sobel

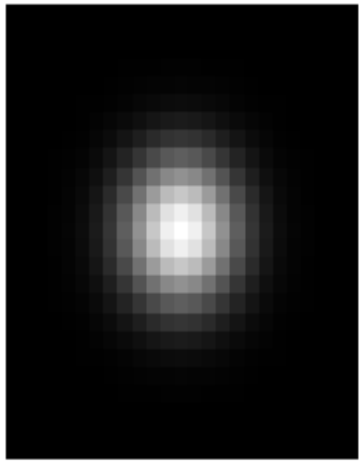


Vertical Edge
(absolute value)

Gaussian Filter

- Weight contributions of neighboring pixels by nearness



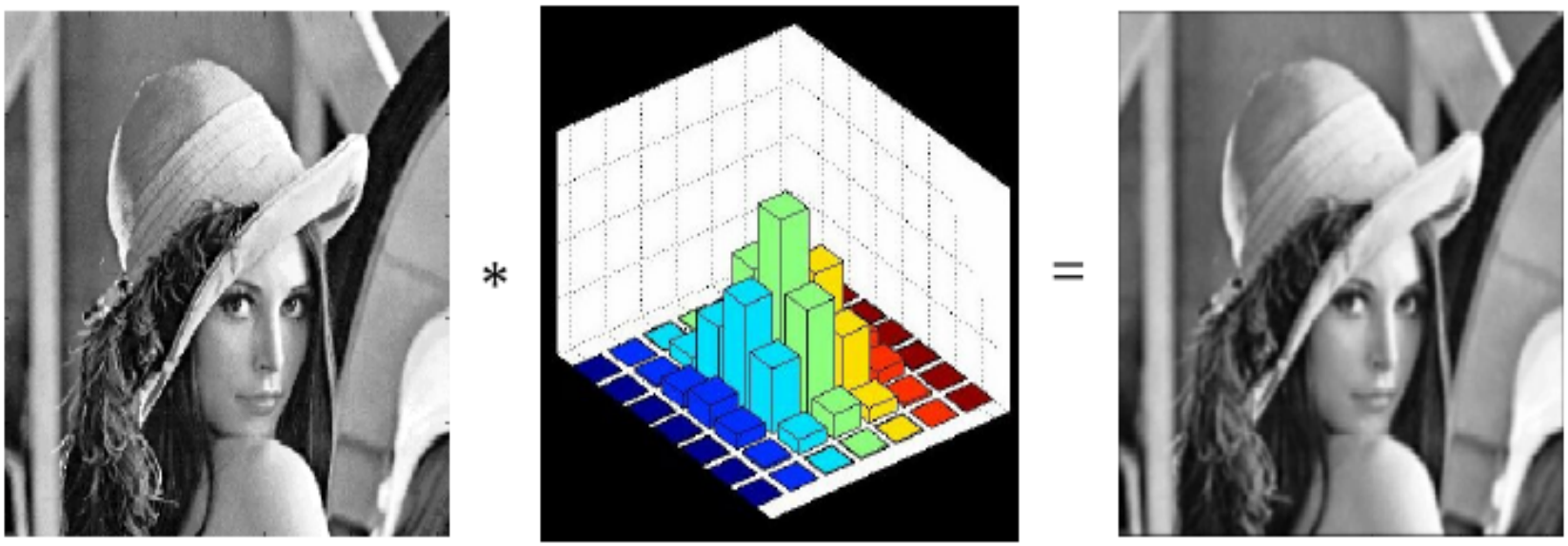


0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

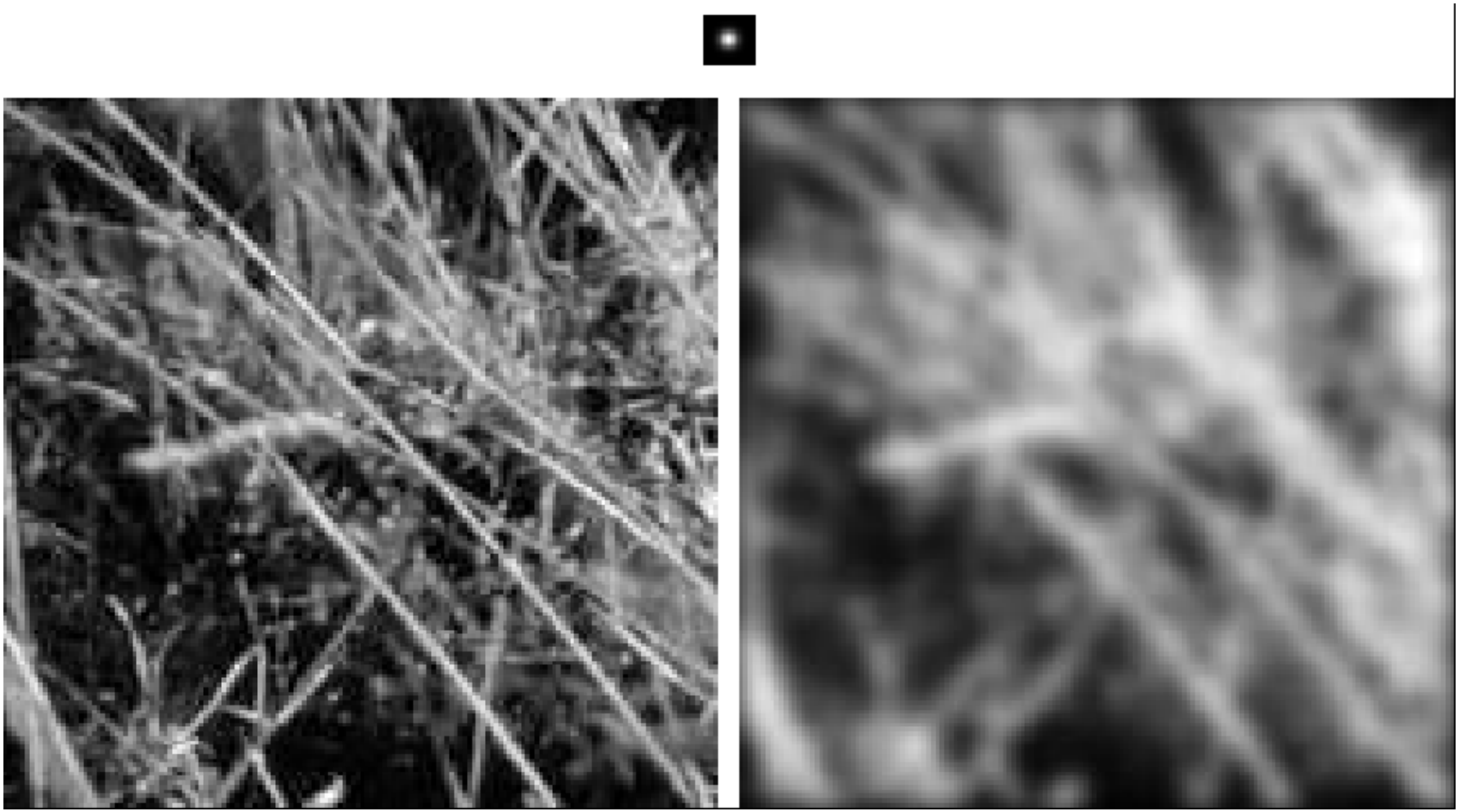
5 x 5, $\sigma = 1$

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

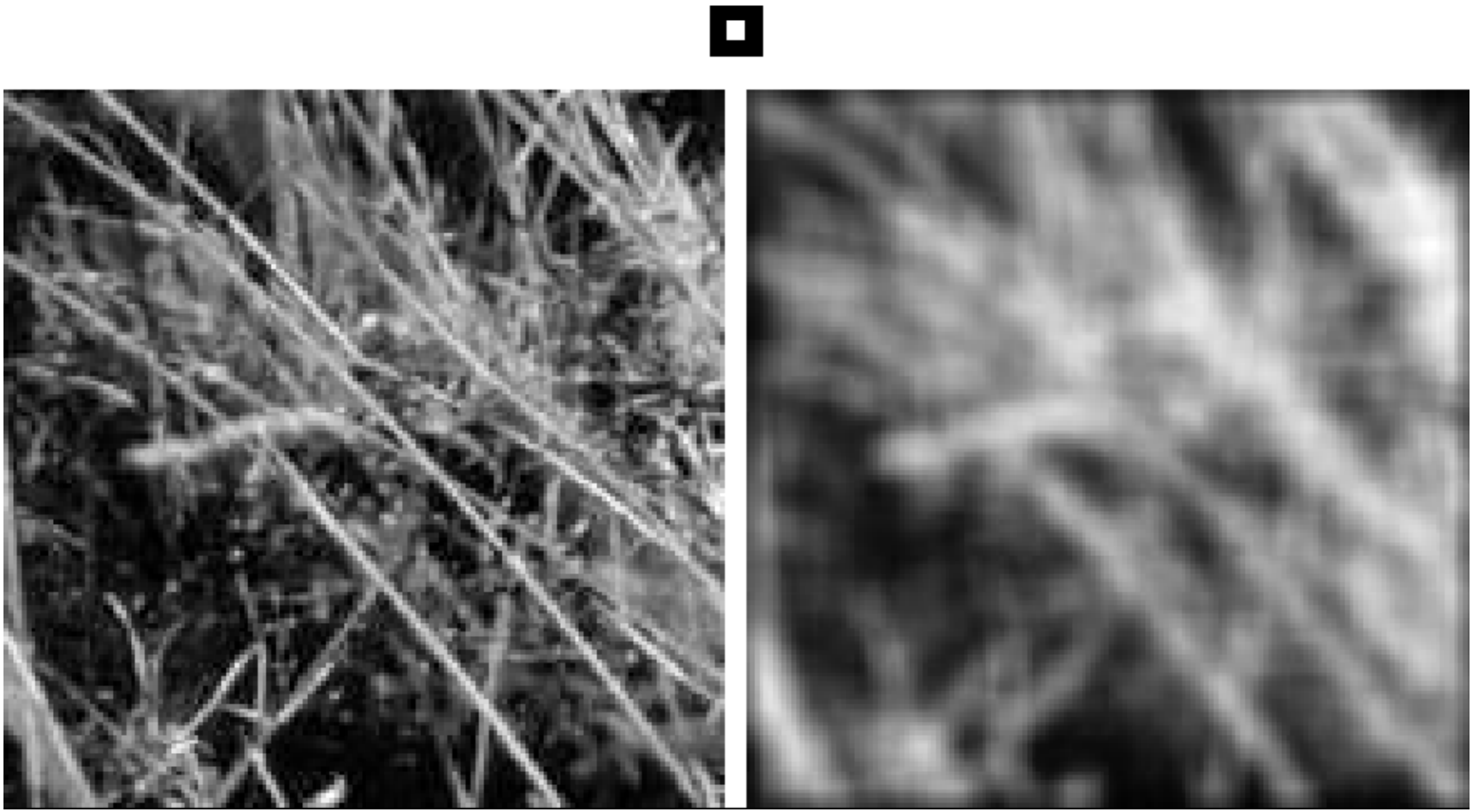
Filtering Gaussian



Smoothing with Gaussian Filter



Smoothing with box Filter



Gaussian Filters

- Remove high-frequency components from the image (low-pass filter): images become more smooth.
- Convolution with self is another Gaussian: So can smooth with small-width kernel, repeat, and get same result as larger-width kernel would have. Convolving two times with Gaussian kernel of width σ is same as convolving once with kernel of width $\sigma\sqrt{2}$.
- Separable kernel: Factors into product of two 1D Gaussians.

Separability of the Gaussian Filter

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2}\exp^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$= \left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{x^2}{2\sigma^2}}\right)\left(\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\frac{y^2}{2\sigma^2}}\right)$$

The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

In this case, the two functions are the (identical) 1D Gaussian

Properties of Gaussian

- Most common natural model.
- Smooth function, it has infinite number of derivatives.
- Fourier Transform of Gaussian is Gaussian.
- Convolution of a Gaussian with itself is a Gaussian.
- There are cells in eye that perform Gaussian filtering.

Outline

Jean Baptiste Joseph Fourier (1768-1830)

had crazy idea (1807):

Any *univariate function can be rewritten as a weighted sum of sines and cosines of different frequencies.*

Jean Baptiste Joseph Fourier (1768-1830)(cont'd)

had crazy idea (1807):

Any *univariate function can be rewritten as a weighted sum of sines and cosines of different frequencies.*

- Don't believe it?
  - Neither did Lagrange, Laplace, Poisson and other big wigs
  - Not translated into English until 1878!

...the manner in which the author arrives at these equations is not exempt of difficulties and...his analysis to integrate them still leaves something to be desired on the score of generality and even rigour.

Thinking in Terms of Frequency

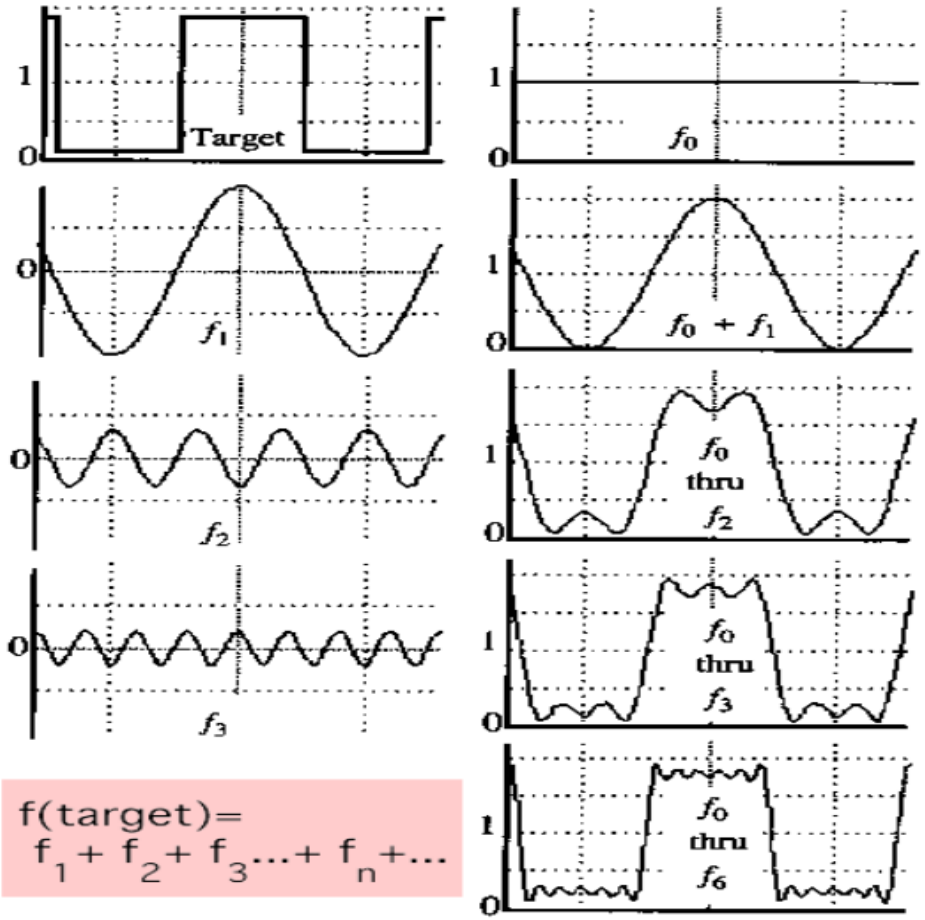
- But it's (mostly) true!
- Called Fourier Series
- There are some subtle restrictions

A Sum of Sines

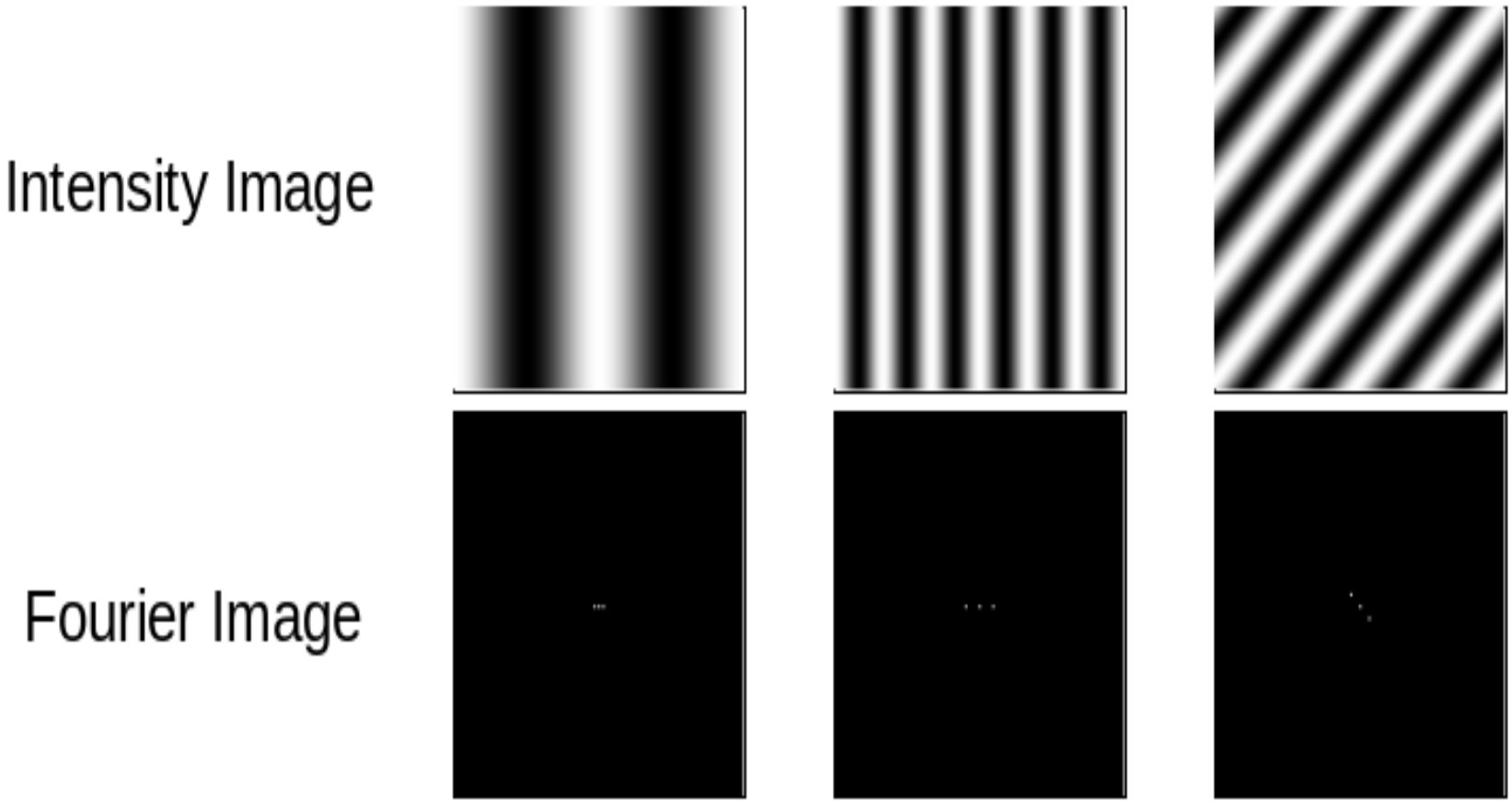
Our building block:

$$A\sin(\omega x+\varphi)$$

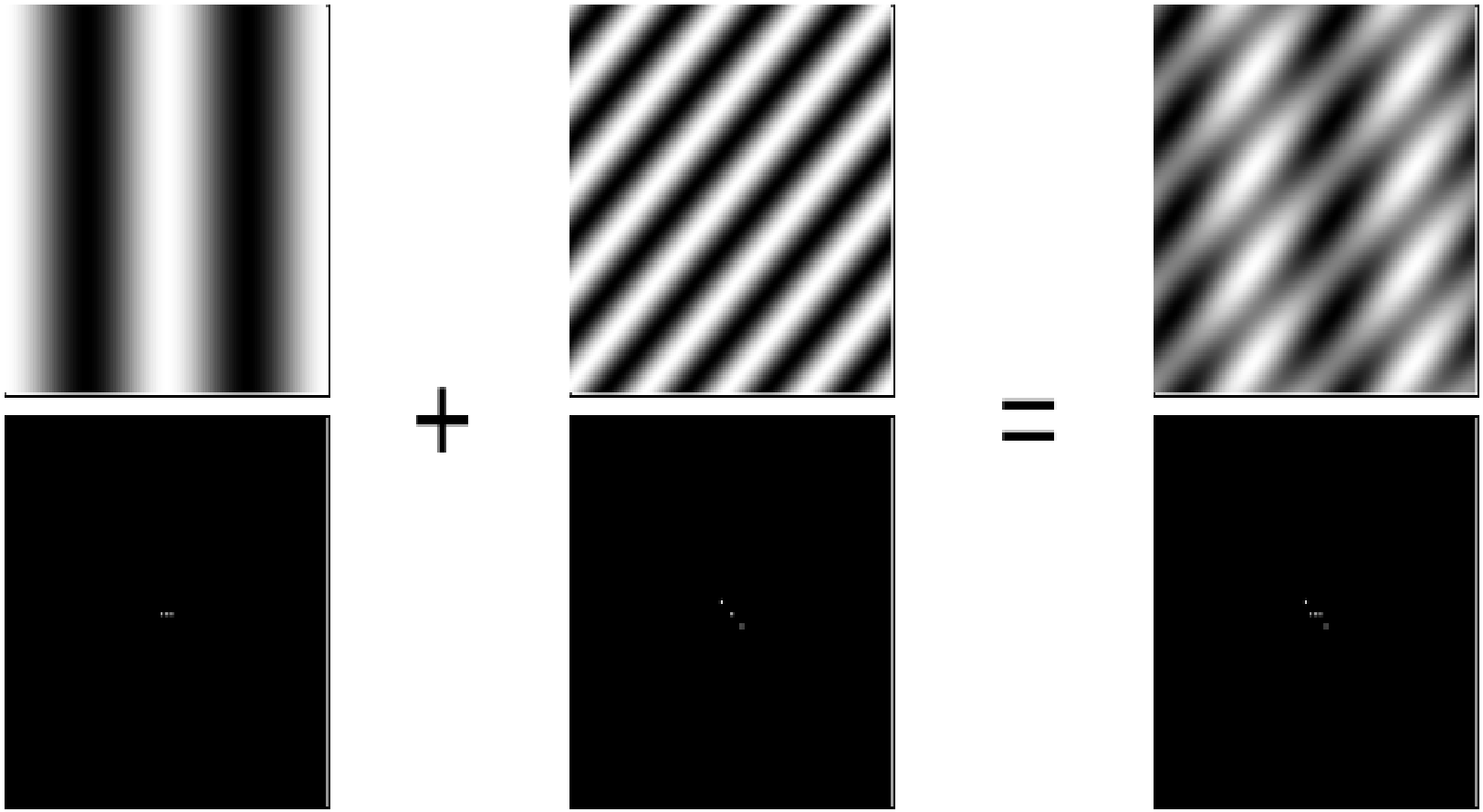
Add enough of them to get any signal *f(x)* you want!



Fourier Analysis in Images



Signals Can Be Composed



Fourier Transform

- Fourier transform stores the magnitude and phase at each frequency
  - Magnitude encodes how much signal there is at a particular frequency
  - Phase encodes spatial information (indirectly)
  - For mathematical convenience, this is often notated in terms of real and complex numbers

Amplitude: $A = \pm\sqrt{R(\omega)^2 + I(\omega)^2}$

Phase: $\phi = \tan^{-1}\frac{I(\omega)}{R(\omega)}$

Computing the Fourier Transform

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

Continuous

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

Discrete

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

k = -N/2..N/2

Fast Fourier Transform (FFT): NlogN

The Convolution Theorem

- The Fourier transform of the convolution of two functions is the product of their Fourier transforms

$$\mathrm{F}[g * h] = \mathrm{F}[g]\mathrm{F}[h]$$

- The inverse Fourier transform of the product of two Fourier transforms is the convolution of the two inverse Fourier transforms

$$\mathrm{F}^{-1}[gh] = \mathrm{F}^{-1}[g] * \mathrm{F}^{-1}[h]$$

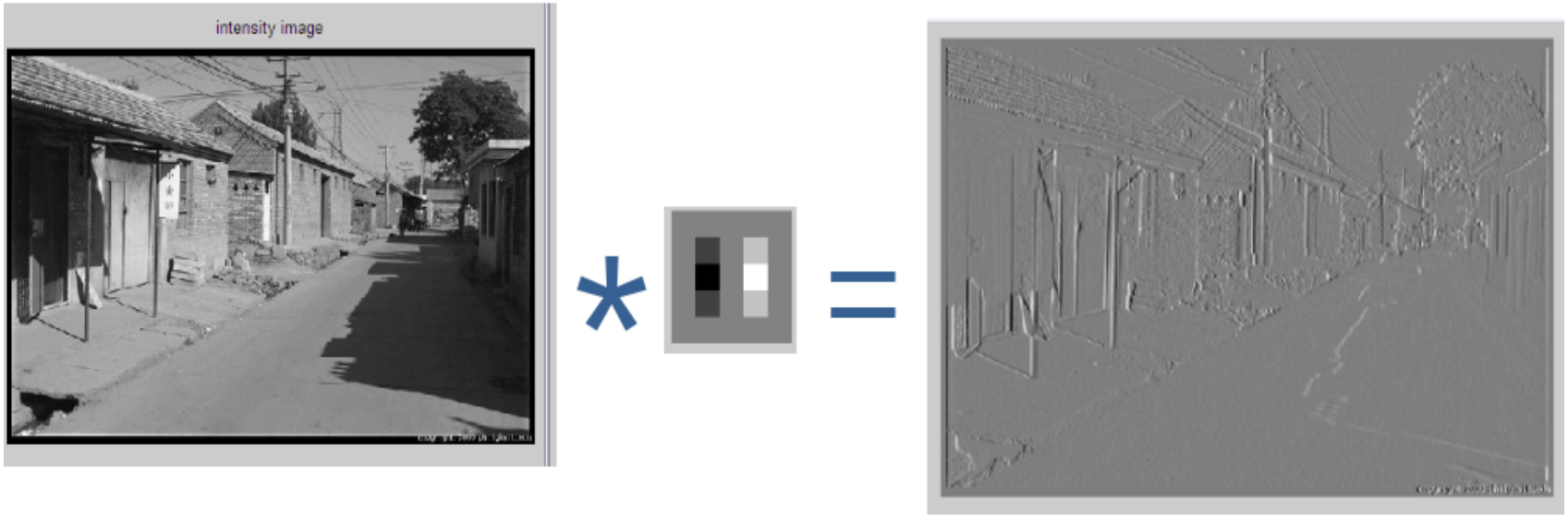
- **Convolution** in spatial domain is equivalent to **multiplication** in frequency domain!

Properties of Fourier Transforms

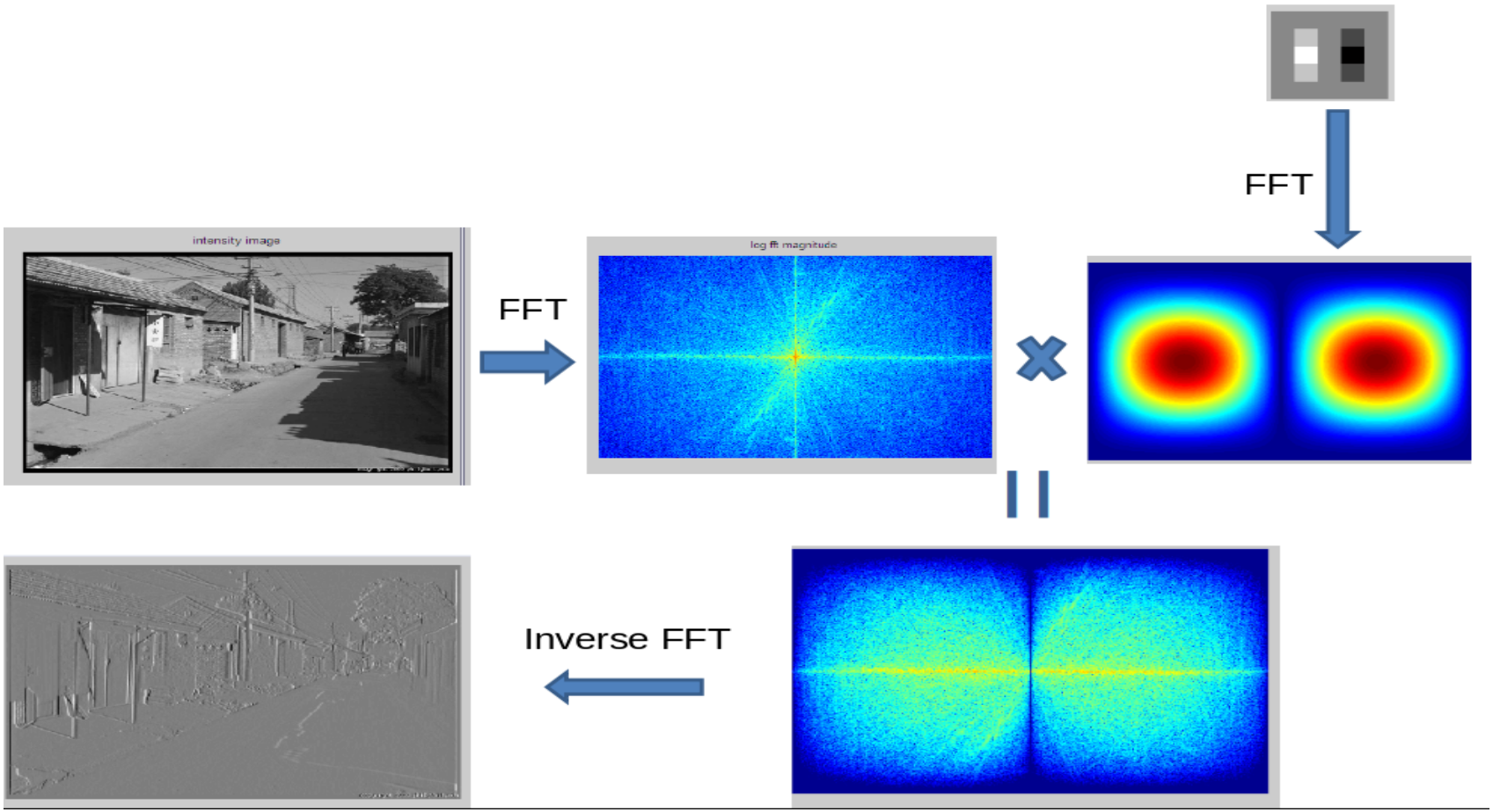
- Linearity $\mathcal{F}[ax(t) + by(t)] = a\mathcal{F}[x(t)] + b\mathcal{F}[y(t)]$
- Fourier transform of a real signal is symmetric about the origin
- The energy of the signal is the same as the energy of its Fourier transform

Filtering in Spatial Domain

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



Filtering in Frequency Domain

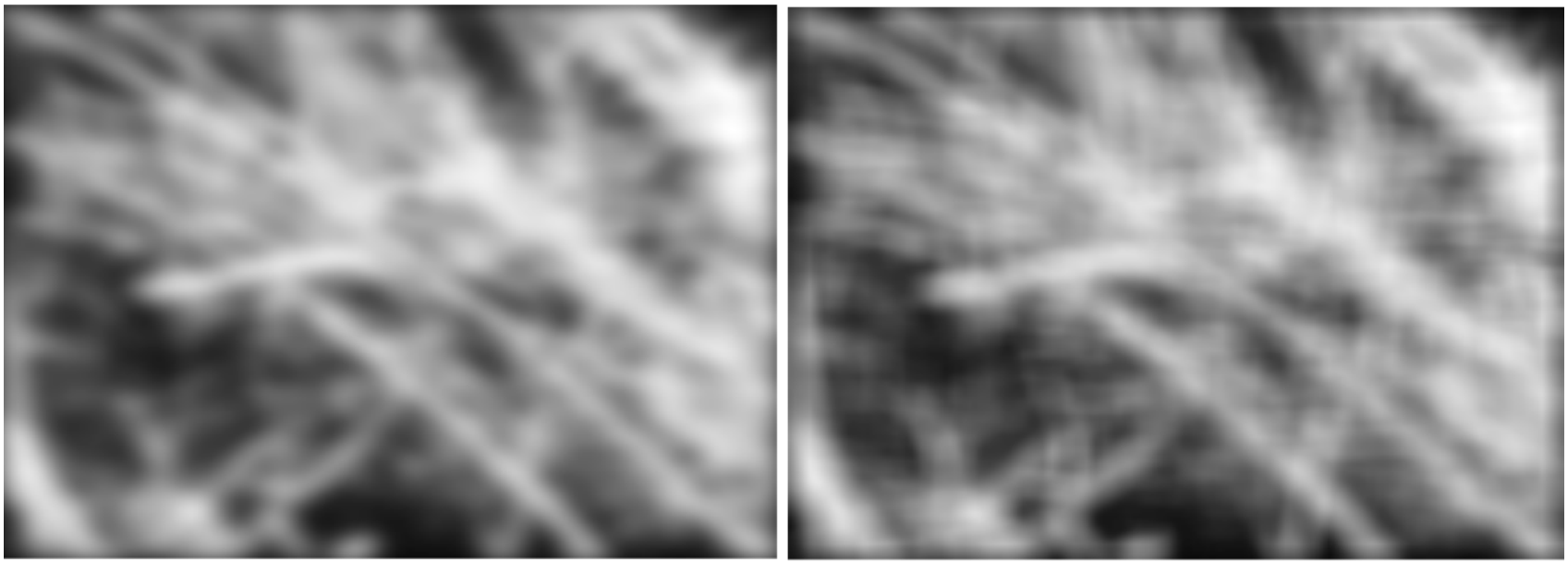


Filtering

Why does the Gaussian give a nice smooth image, but the square filter give edgy artifacts?

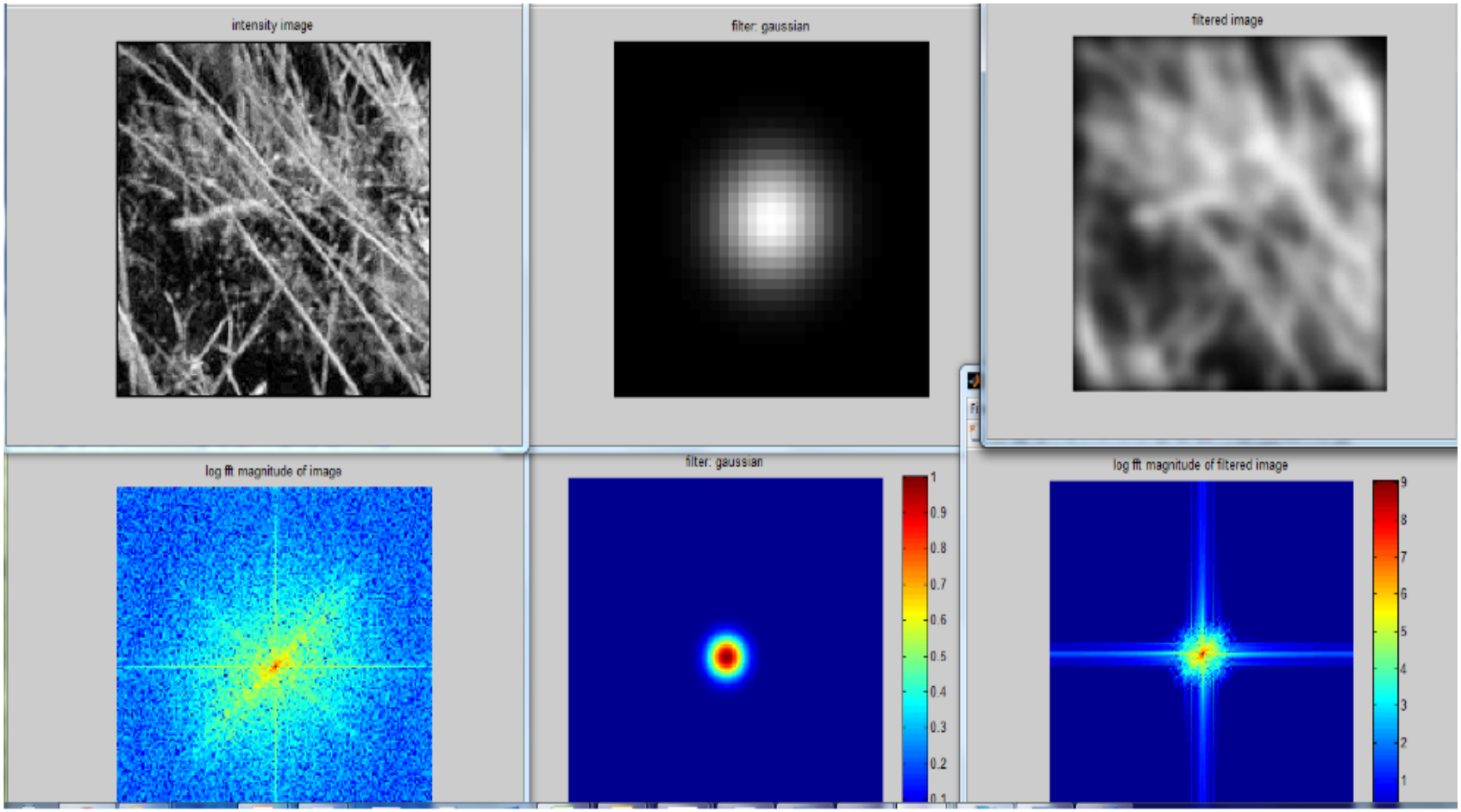
Gaussian 

Box filter 

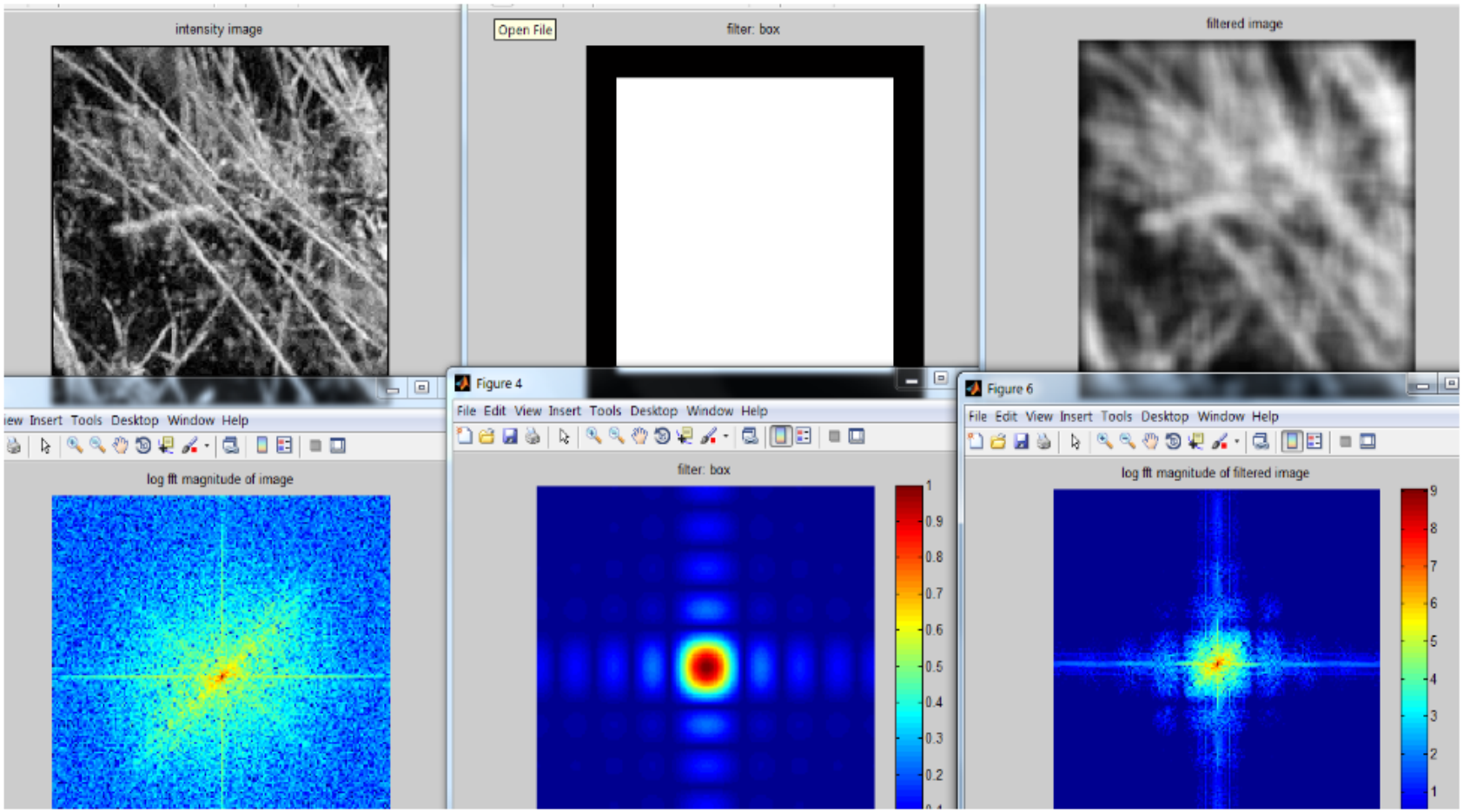


Things we can't understand without thinking in frequency

Filtering: Gaussian

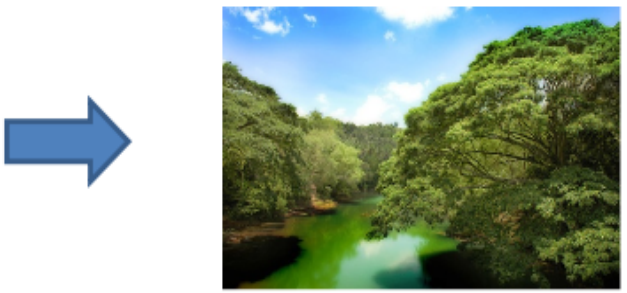


Filtering: Box Filter

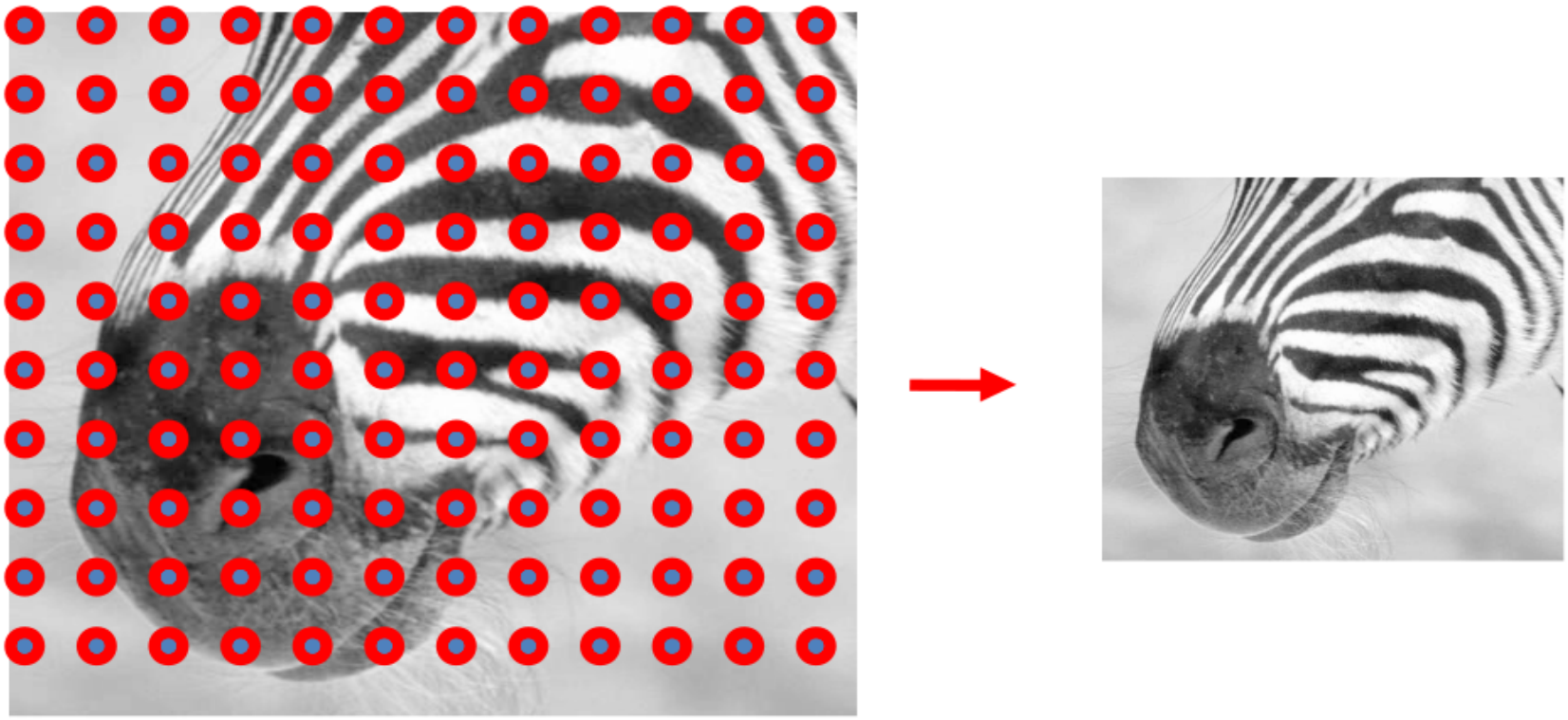


Sampling

Why does a lower resolution image still make sense to us? What do we lose?



Subsampling by a factor of 2



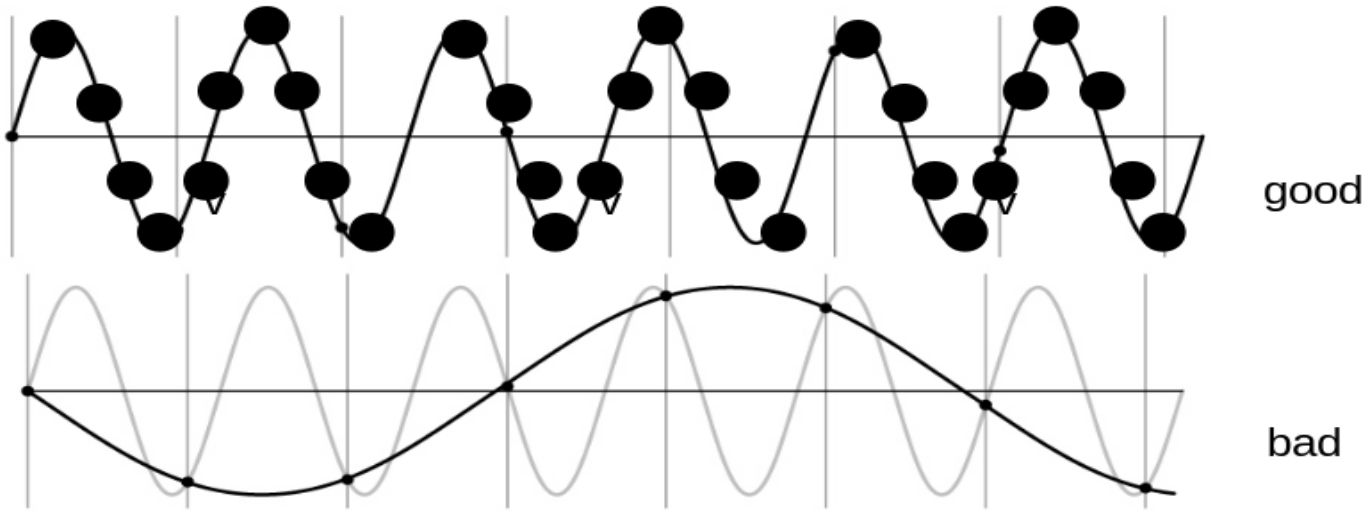
Throw away every other row and column to create a 1/2 size image

Aliasing Problem

- Sub-sampling may be dangerous.
- Characteristic errors may appear:
  - Wagon wheels rolling the wrong way in movies.
  - Checkerboards disintegrate in ray tracing.
  - Striped shirts look funny on color television.

Nyquist-Shannon Sampling Theorem

- When sampling a signal at discrete intervals, the sampling frequency must be $\geq 2 \times f_{max}$
- f_{max} = max frequency of the input signal
- This will allows to reconstruct the original perfectly from the sampled version



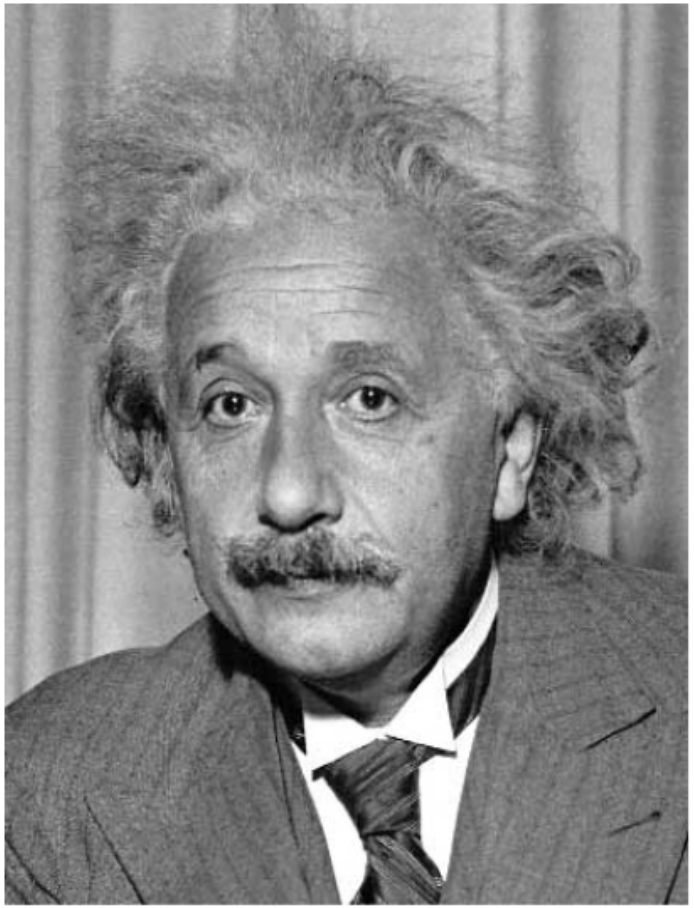
Anti-aliasing

- Solutions: Sample more often.
- Get rid of all frequencies that are greater than half the new sampling frequency:
  - Will lose information.
  - But it's better than aliasing.
  - Apply a smoothing filter.

Outline

Template Matching

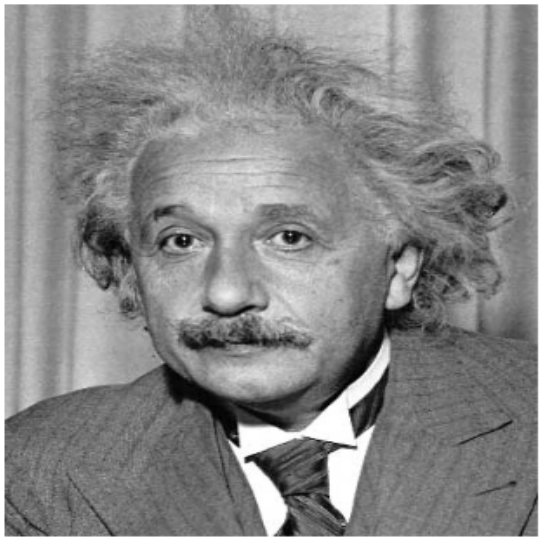
- Goal: find  in image
- Main challenge: What is a good similarity or distance measure between two patches?
  - Correlation
  - Zero-mean correlation
  - Sum Square Difference
  - Normalized Cross Correlation



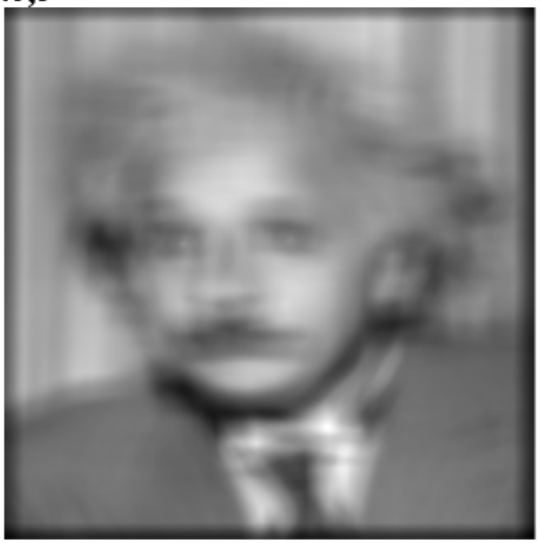
Matching with Filters

- Goal: find  in image
- **Method 0**: filter the image with eye patch $h[m,n] = \sum_{k,l} g[k,l]\ f[m+k,n+l]$

f = image
g = filter



Input



Filtered Image

What went wrong?

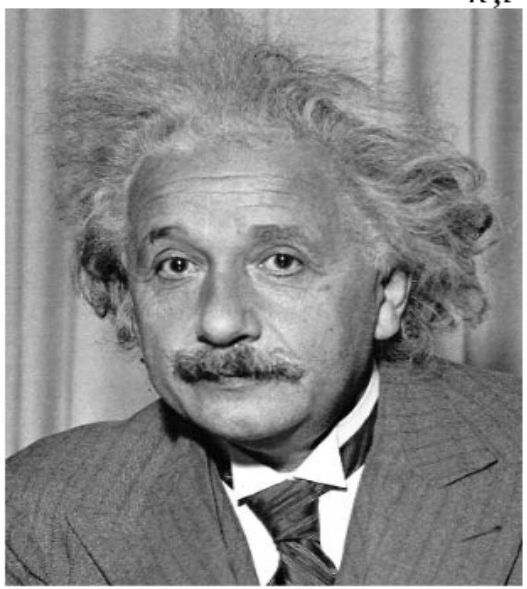
Problem: response is stronger for higher intensity

Matching with Filters (cont'd)

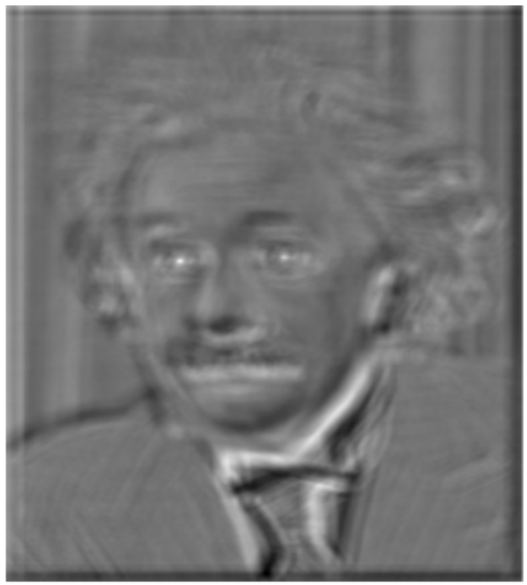
- Goal: find  in image
- **Method 1**: filter the image with zero-mean eye

$$h[m,n] = \sum_{k,l} (f[k,l] - \bar{f})(g[m+k,n+l])$$

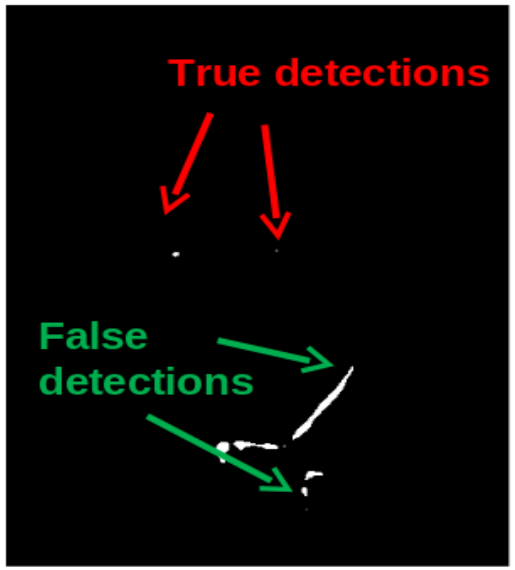
mean of f



Input



Filtered Image (scaled)



Thresholded Image

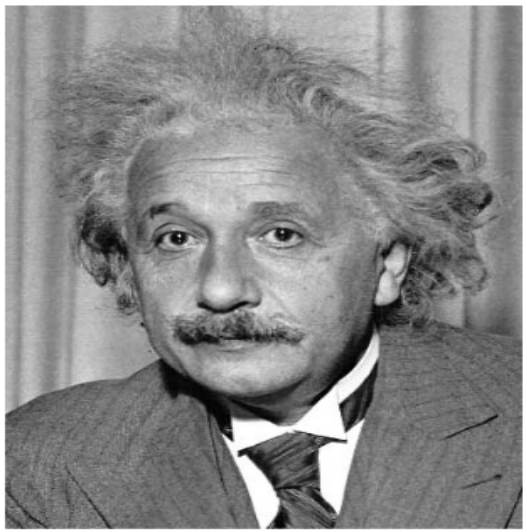
Matching with Filters (cont'd)

- Likes bright pixels where filters are above average, dark pixels where filters are below average.
- Problems: response is sensitive to gain/contrast, pixels in filter that are near the mean have little effect, does not require pixel values in image to be near or proportional to values in filter.

Matching with Filters (cont'd)

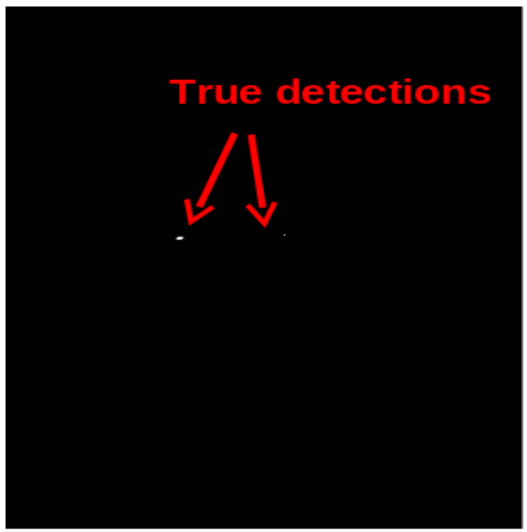
- Goal: find  in image
- Method 2: SSD

$$h[m,n] = \sum_{k,l} (g[k,l] - f[m+k,n+l])^2$$



Input

1- sqrt(SSD)



Thresholded Image

Problem: SSD sensitive to average intensity

Matching with Filters (cont'd)

- Goal: find  in image
- Method 3: Normalized cross-correlation

mean template

mean image patch

$$h[m,n] = \frac{\sum_{k,l}(g[k,l]-\bar{g})(f[m-k,n-l]-\bar{f}_{m,n})}{\left(\sum_{k,l}(g[k,l]-\bar{g})^2 \sum_{k,l}(f[m-k,n-l]-\bar{f}_{m,n})^2\right)^{0.5}}$$

Invariant to mean and scale of intensity

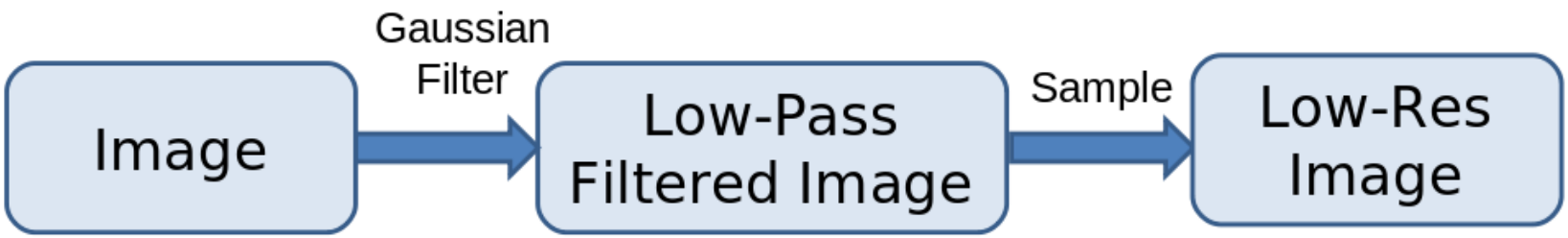
Q: What is the best method to use?

- A: Depends
- SSD: faster, sensitive to overall intensity.
- Normalized cross-correlation: slower, invariant to local average intensity and contrast.

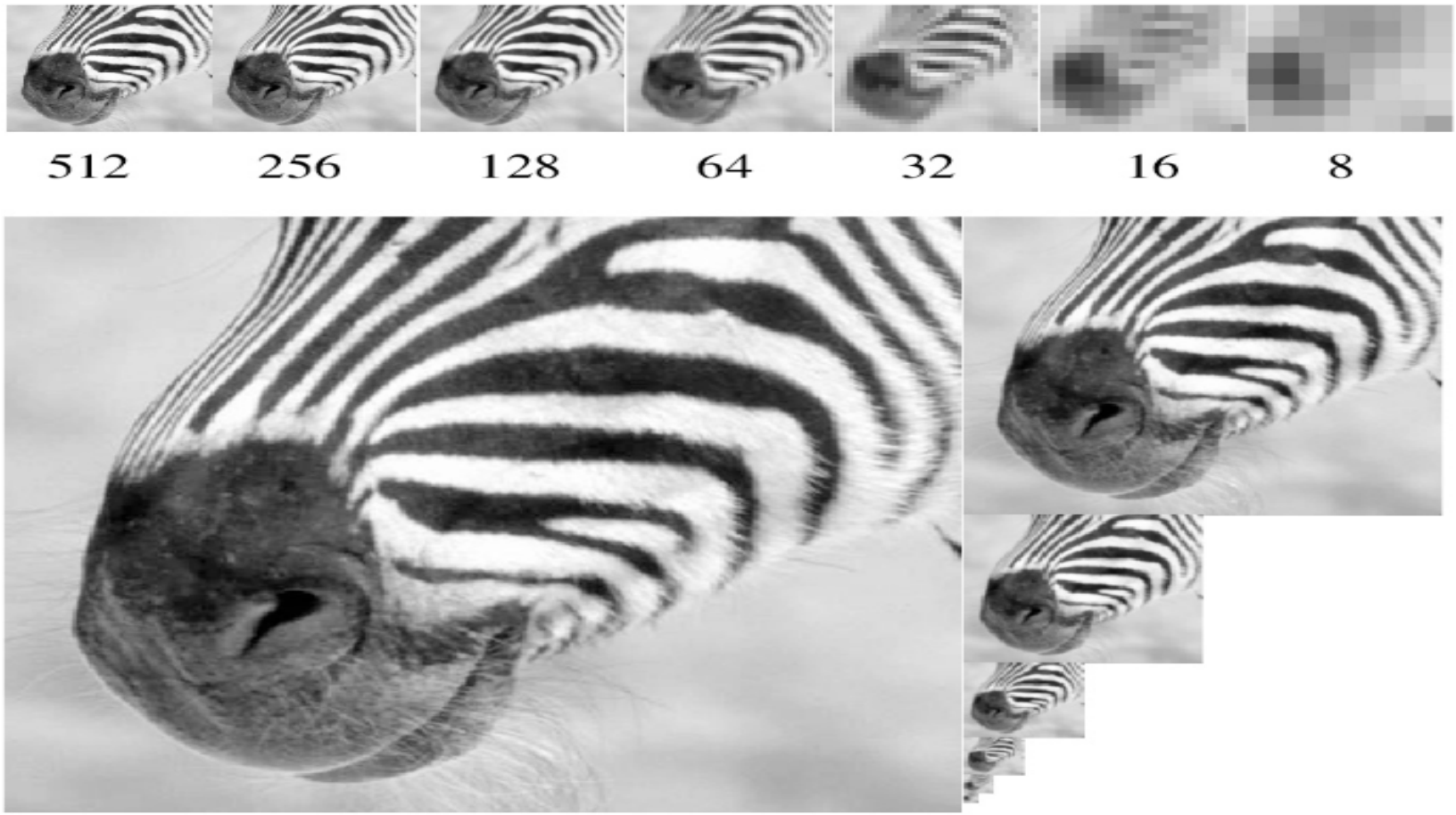
Q: What if we want to find larger or smaller eyes?

- A: Image Pyramid

Review of Sampling



Gaussian Pyramid



Template Matching with Image Pyramids

Input: Image, Template

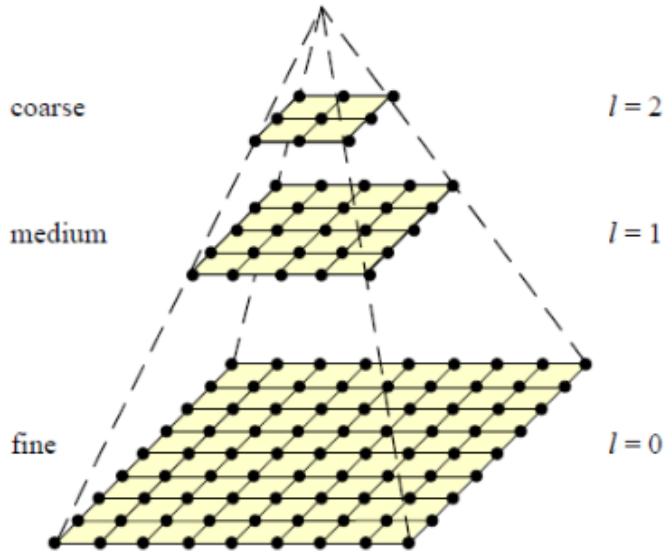
1. Match template at current scale.
2. Downsample image.
3. Repeat 1-2 until image is very small.
4. Take responses above some threshold, perhaps with non-maxima suppression.

Coarse-to-fine Image Registration

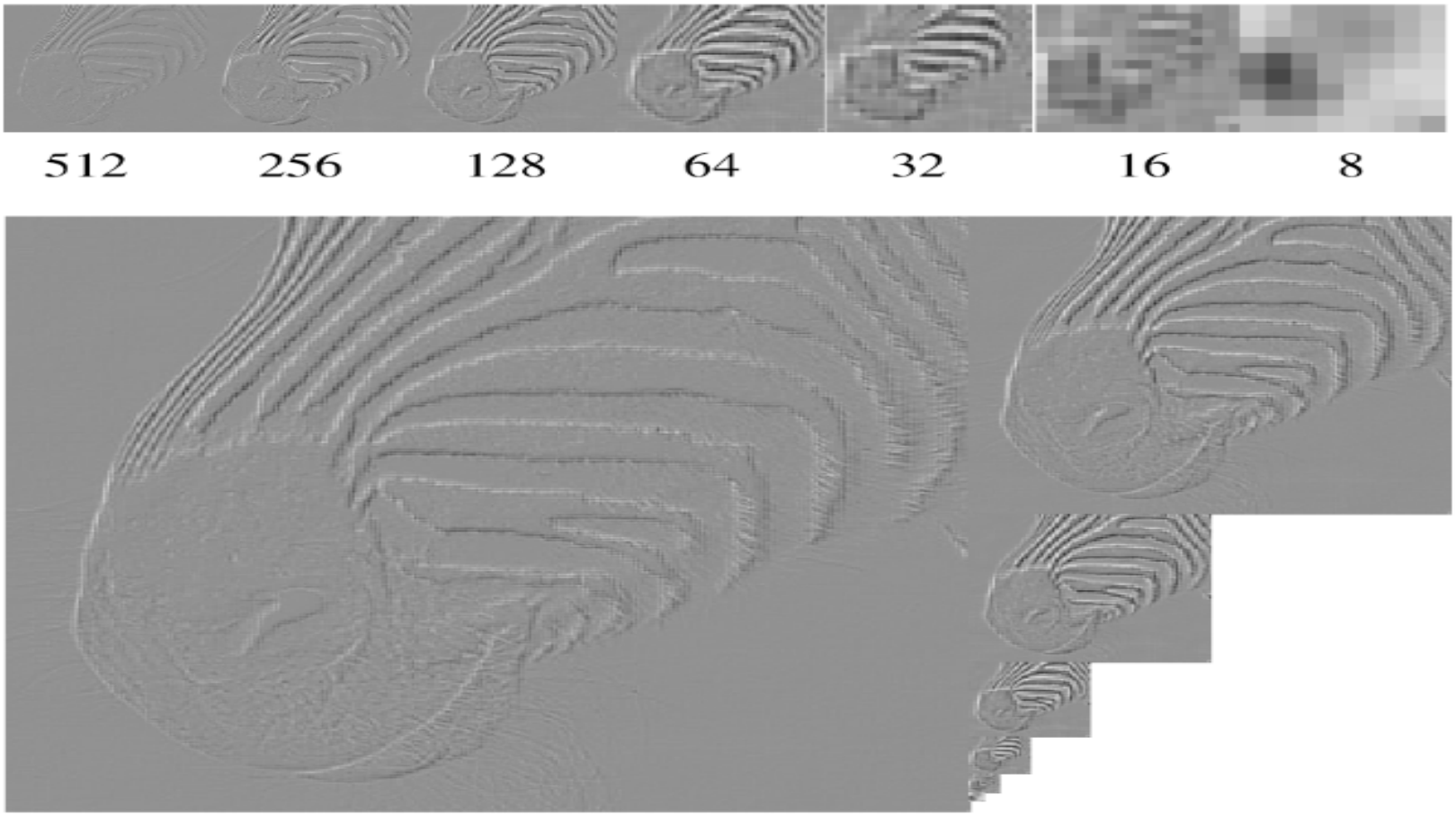
1. Compute Gaussian pyramid
2. Align with coarse pyramid
3. Successively align with finer pyramids
   - Search smaller range

Why is this faster?

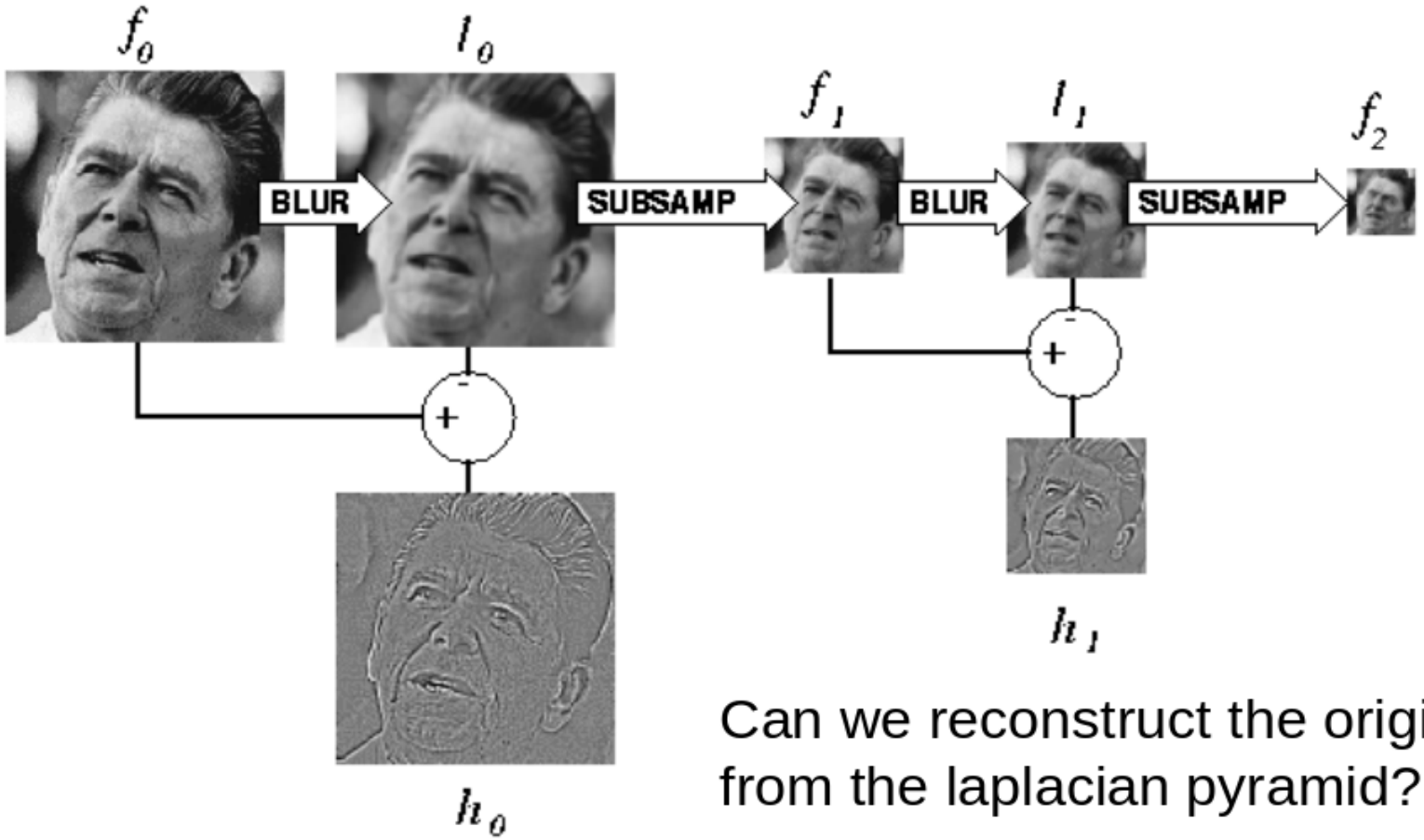
Are we guaranteed to get the same result?



Laplacian Pyramid



Computing Gaussian/Laplacian Pyramid



Can we reconstruct the original from the laplacian pyramid?

Hybrid Image

Hybrid Image in Laplacian Pyramid

High frequency → Low frequency

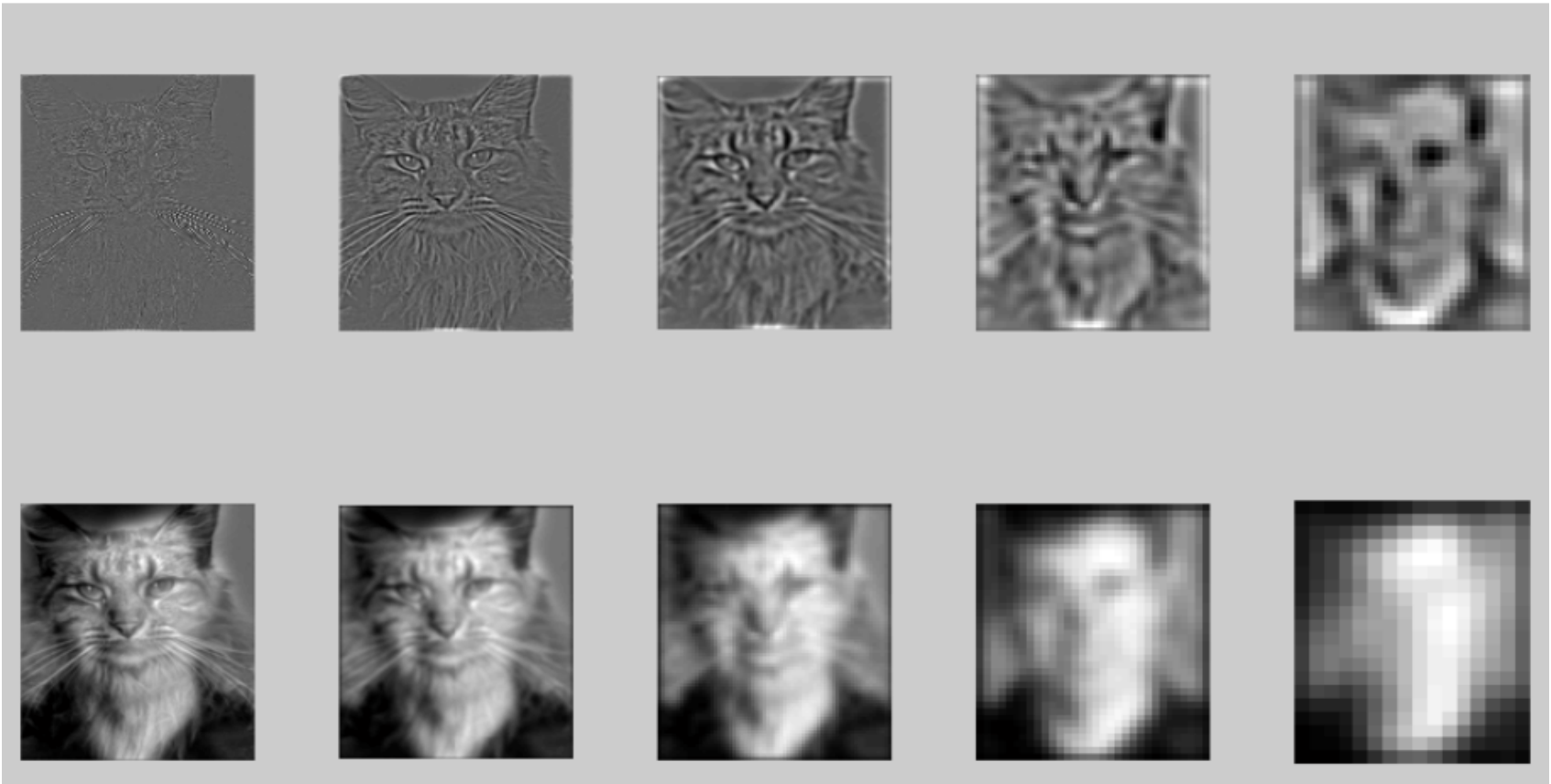


Image Representation

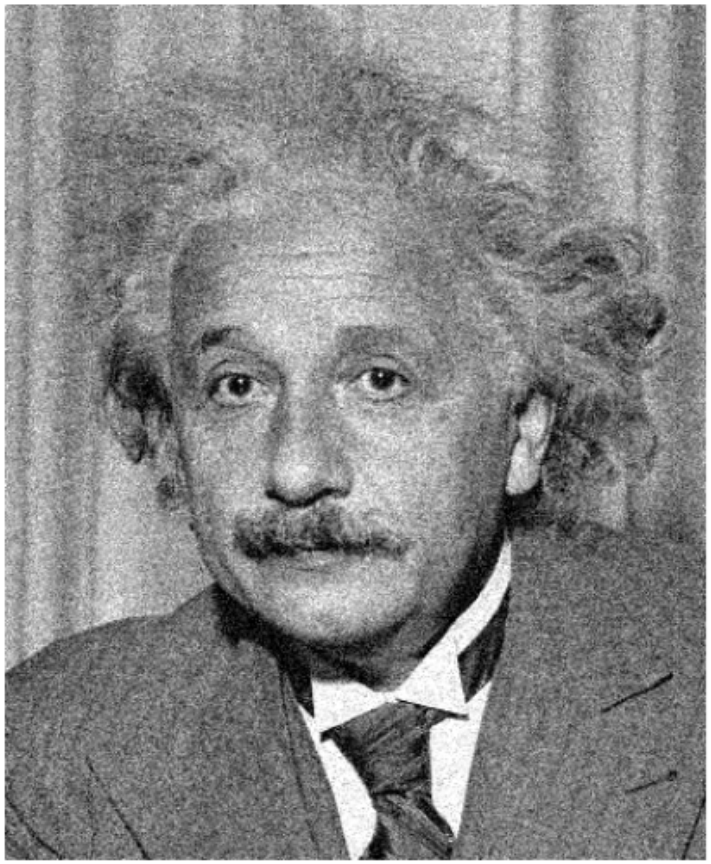
- Pixels: great for spatial resolution, poor access to frequency.
- Fourier transform: great for frequency, not for spatial info.
- Pyramids/filter banks: balance between spatial and frequency information.

Major Uses of Image Pyramids

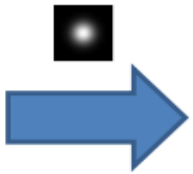
- Compression.
- Object detection: Scale search and Features.
- Detecting stable interest points.
- Registration: Course-to-fine.

Outline

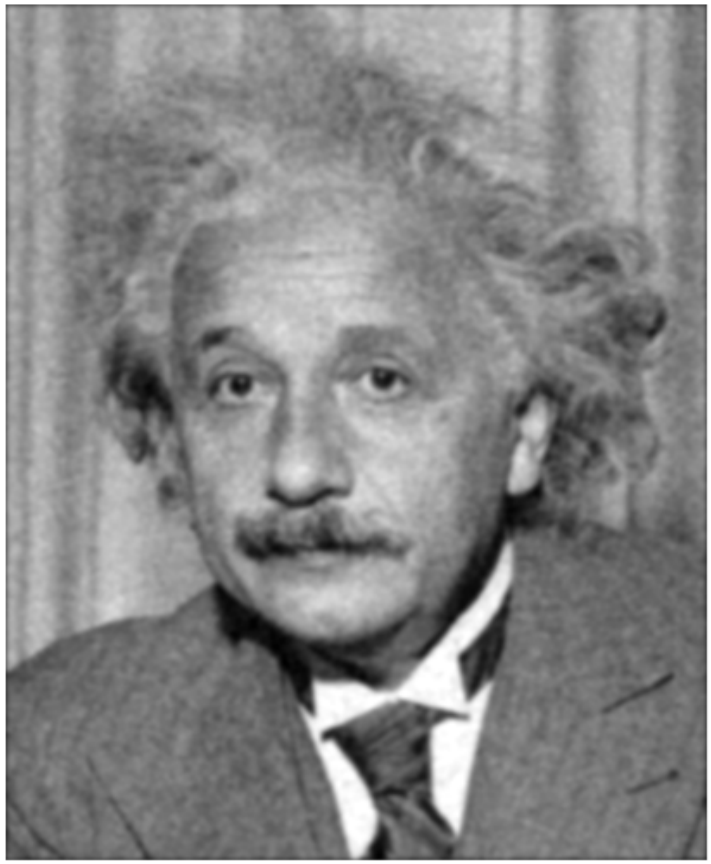
Denoising



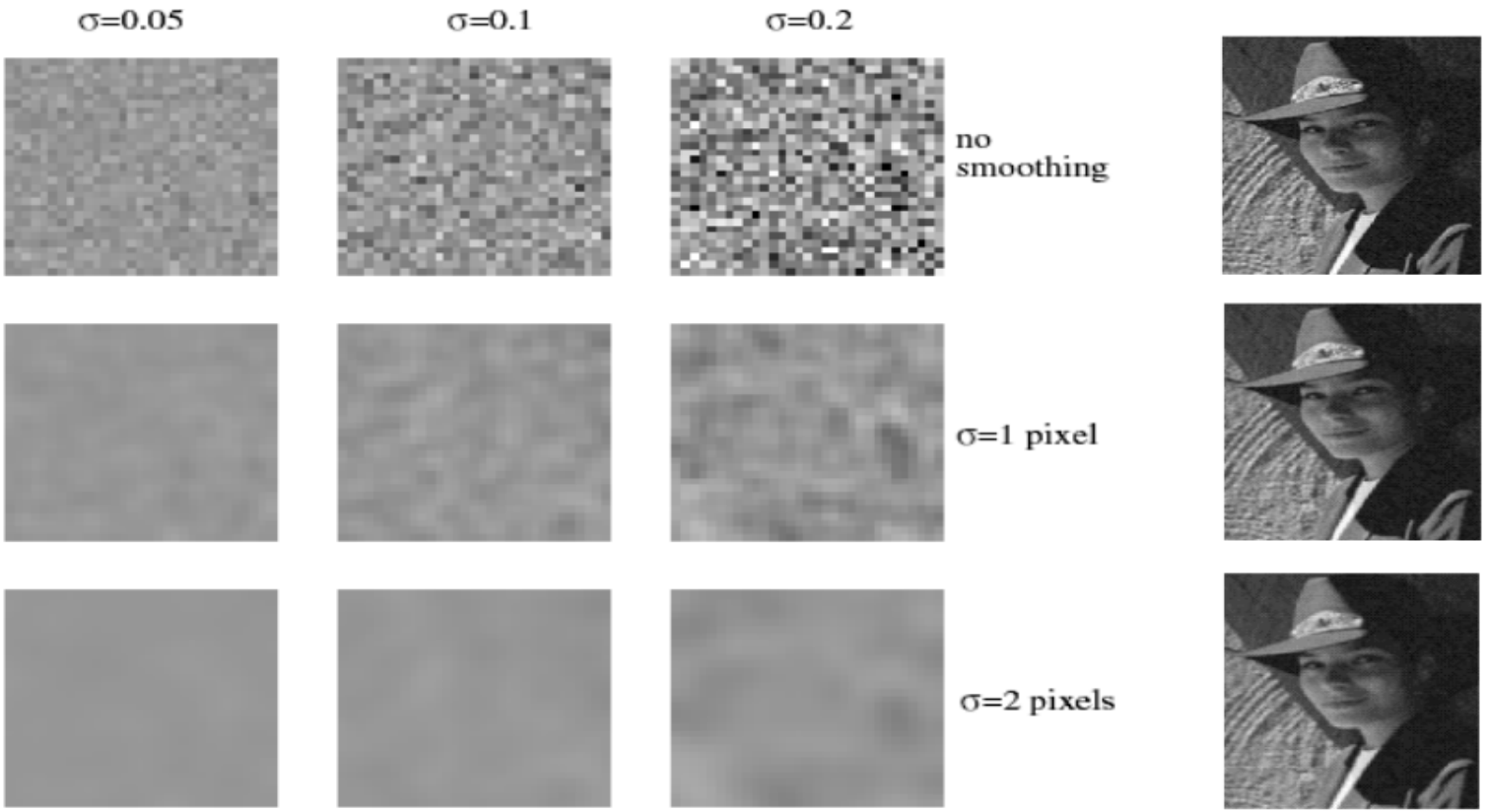
Additive Gaussian Noise



Gaussian Filter



Reducing Gaussian Noise

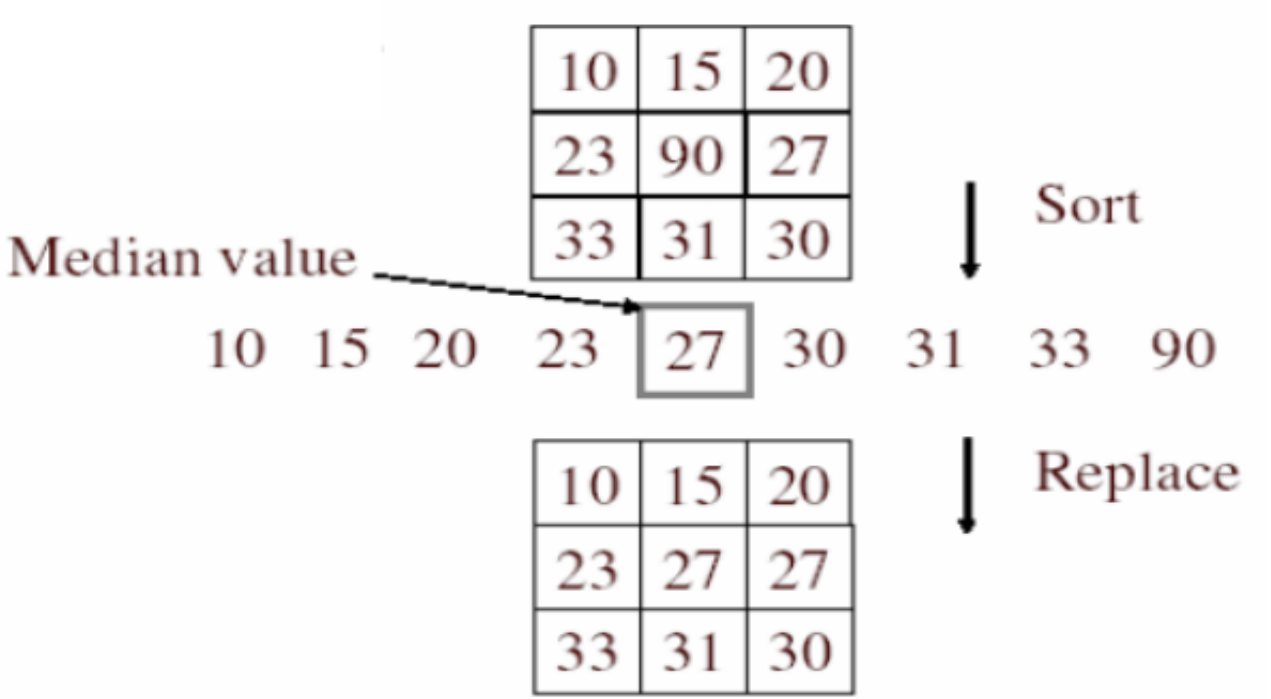


Smoothing with larger standard deviations suppresses noise, but also blurs the image

Reducing Salt-and-Pepper Noise by Gaussian Smoothing

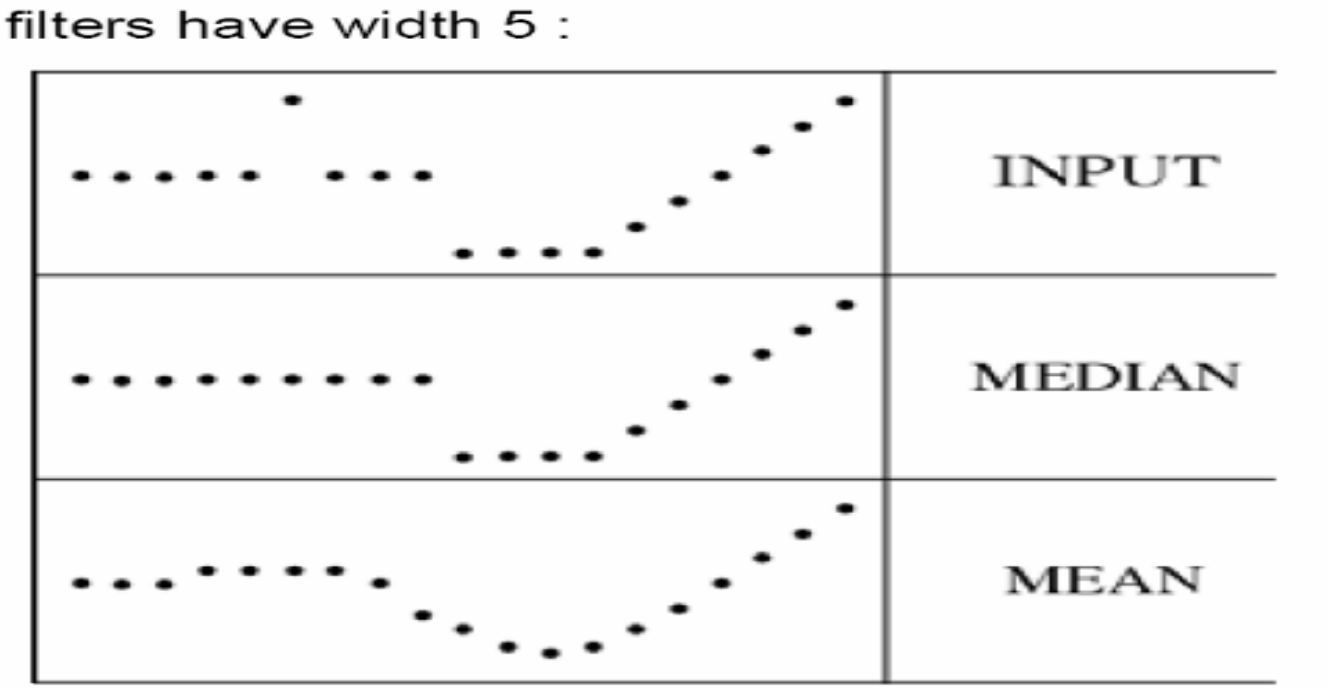
Alternative idea: Median filtering

A median filter operates over a window by selecting the median intensity in the window. **Better at salt and pepper noise**. Not convolution. **Is median filtering linear?**

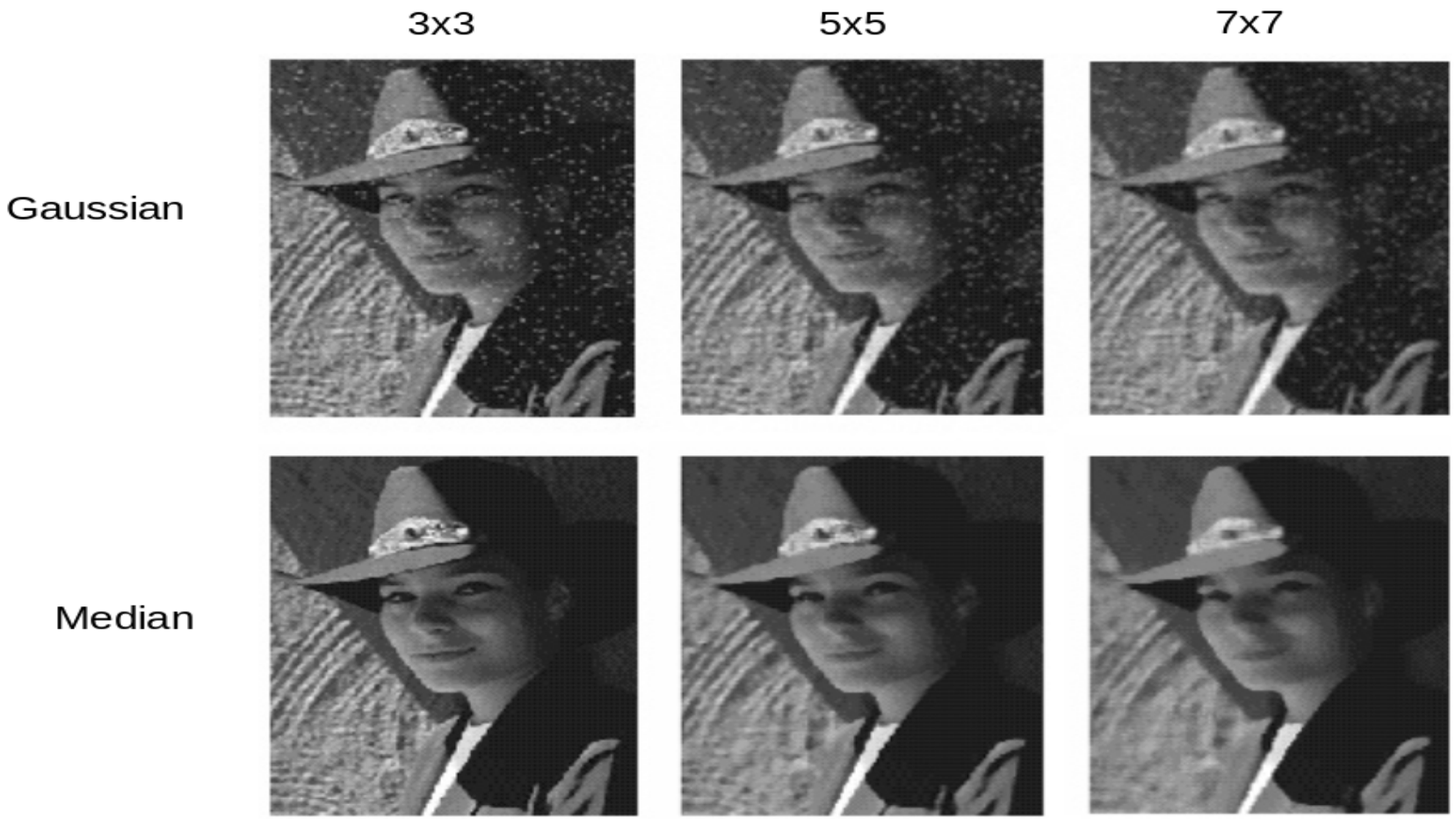


Median Filter

What advantage does median filtering have over Gaussian filtering? Robustness to outliers



Median vs. Gaussian Filtering



Other Non-linear Filters

- Weighted median:pixels further from center count less.
- Clipped mean:average, ignoring few brightest and darkest pixels.
- Bilateral filtering: weight by spatial distance and intensity difference.

Bilateral Filtering

References

- Richard Szeliski, Computer Vision: Algorithms and Applications, 2nd Ed., Springer-Verlag London, 2020.
- Reinhard Klette, Concise Computer Vision: An Introduction into Theory and Algorithms, Springer-Verlag London 2014.
- Mubarak Shah, CV Course: Lecture 2 (http://www.youtube.com/watch?v=715uLCHt4jE&feature=plcp).
- CSCI 1430: Introduction to Computer Vision, (https://cs.brown.edu/courses/csci1430/)
- Deep Learning for Computer Vision, University of Michigan (https://web.eecs.umich.edu/ justincj/teaching/eecs498/FA2020/sylla