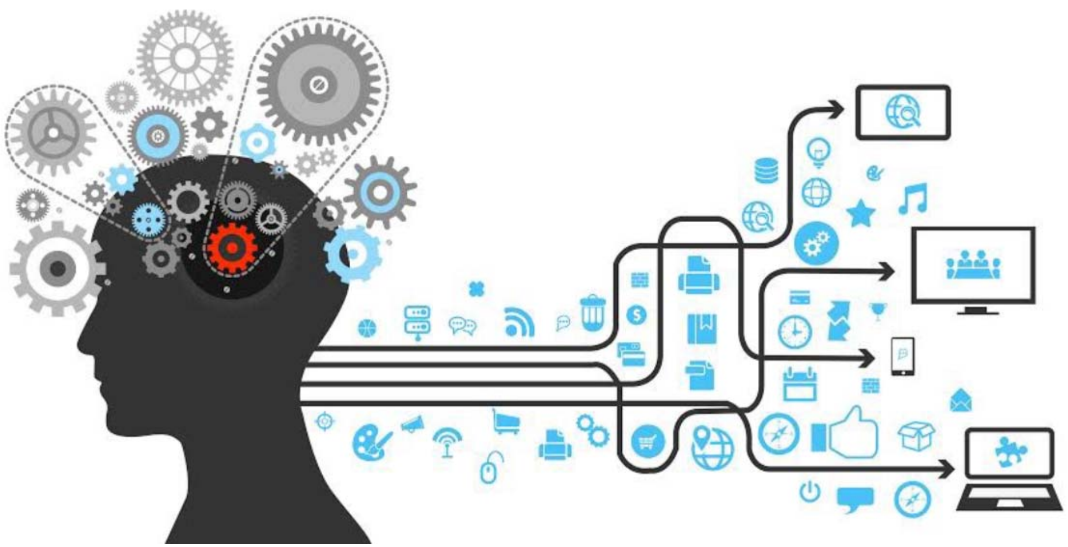
Computer Vision

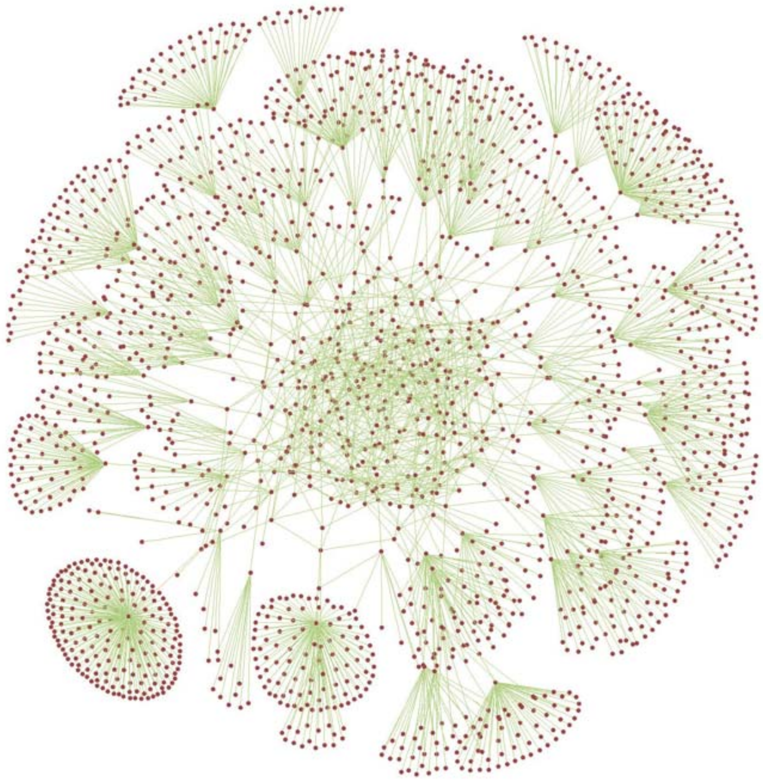
Early vision: Just one image

School of Electronic &
Electrical Engineering

Sungkyunkwan University

Hyunjin Park





Linear Filters

Representing image regions

- We can generate a number of intrinsic images from a given image
  - Depth/disparity
  - Surface albedo/color
  - Surface normal
  - …

- How can we organize these into surfaces?
  - Identify attributes of regions
    - Bounding edges
    - Texture
  - Spatial aggregations of pixels
    - Segmentation

Taxonomy

- Signal processing
  - Discrete-time signal processing
  - Wavelet tour of signal processing
- Image processing
  - Two-dimensional signal and image processing
  - The Fourier transform and its applications
- Tools which have become indispensable to computer vision
  - Linear filters
  - Over-complete (pyramid) representation

Systems and filters

- Filtering
  - ***Form a new image*** whose pixels are a combination of original pixel values

- Goals
  - Extract useful information from the images
    - Features (edges, corners, blobs, …)
  - Modify or enhance image properties
    - Super-resolution, in-painting, de-nosing

Systems and filters

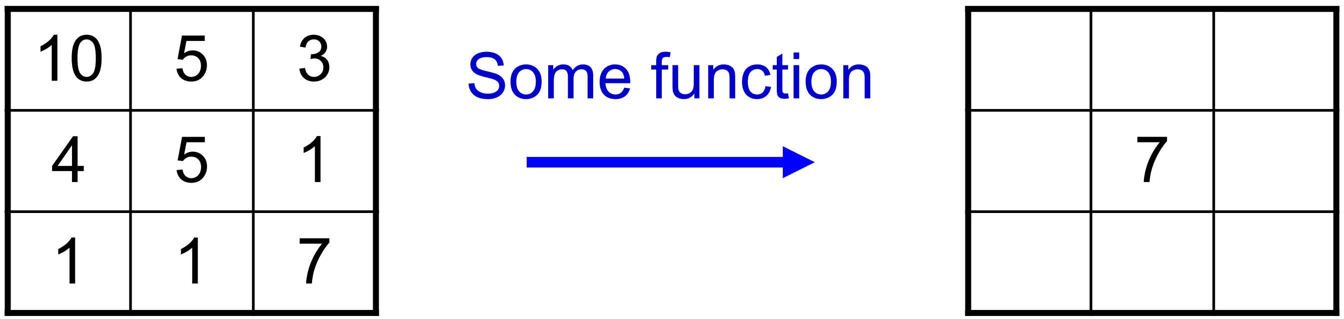
De-noising

Super-resolution

In-painting

Image filtering

- Filtering
  - Modify the pixels in an image
  - Based on some function of a local neighborhood of the pixels



Linear filtering

- Linear case is the simplest and most useful
  - Form a new image by replacing each pixel with a weighted sum (i.e., linear combination) of its neighbors, using the same set of weights at each point

- The prescription for the linear combination is called the ***kernel***

$$\begin{array}{|c|c|c|}\hline 10 & 5 & 3 \\ \hline 4 & 5 & 1 \\ \hline 1 & 1 & 7 \\ \hline \end{array} \ast \underbrace{\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0.5 & 0 \\ \hline 0 & 1.0 & 0.5 \\ \hline \end{array}}_{\text{kernel}} = \begin{array}{|c|c|c|}\hline & & \\ \hline & 7 & \\ \hline & & \\ \hline \end{array}$$

Moving average

- Example: 2D discrete-space moving average with 3×3 window

$$g[n,m] = \frac{1}{9}\sum_{k=n-1}^{n+1}\sum_{l=m-1}^{m+1} f[k,l]$$

$$= \frac{1}{9}\sum_{k=-1}^{1}\sum_{l=-1}^{1} f[n-k,m-l]$$

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

$$(f * h)[m,n] = \frac{1}{9}\sum_{k,l} f[k,l]h[m-k,n-l]$$

Moving average

$F[x,y]$

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

$G[x,y]$

	0								

Moving average

$F[x, y]$

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

$G[x, y]$

	0	10							

Moving average

$F[x, y]$

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

$G[x, y]$

	0	10	20						

Moving average

$F[x,y]$

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

$G[x,y]$

	0	10	20	30					

Moving average

$F[x, y]$

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

$G[x, y]$

	0	10	20	30	30				

Moving average

$F[x, y]$

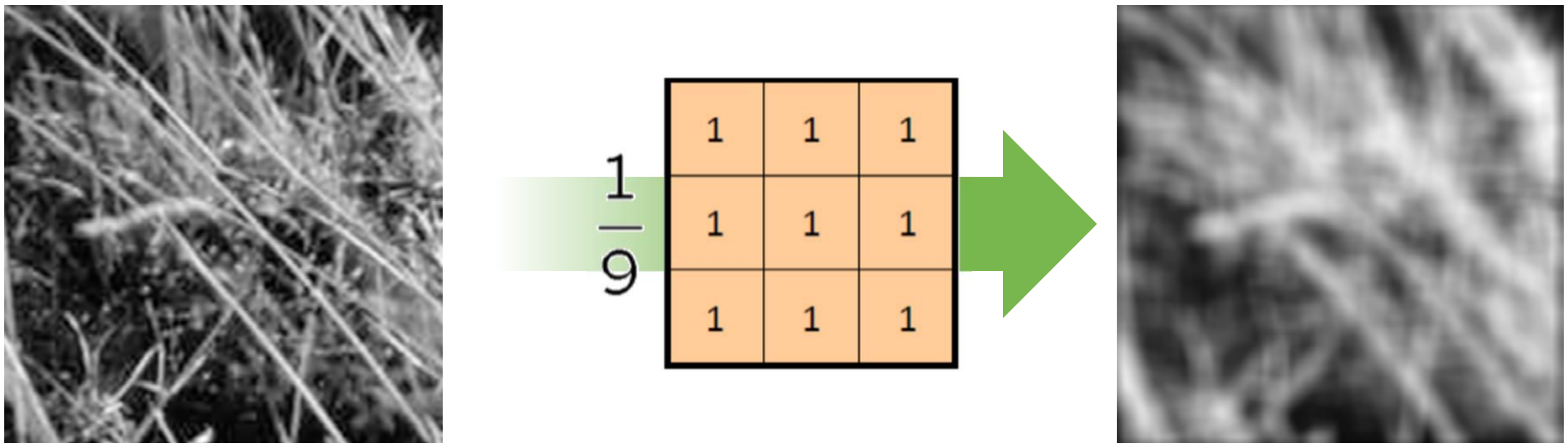
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	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	

Moving average

- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect
  - Remove sharp features



Linear filter properties

- Linear
  - Output is a linear function of the input
    - Superposition: $h * (f_1 + f_2) = (h * f_1) + (h * f_2)$
    - Scaling: $h * (kf) = k(h * f)$
    - $S[\alpha f_1 + \beta f_2] = \alpha S[f_1] + \beta S[f_2]$

Linear filter properties

- Shift-invariant
  - Output is a shift-invariant function of the input
    - Shift the input image two pixels to the left, the output is shifted two pixels to the left
    - If $f[n,m] \xrightarrow{S} g[n,m]$, then $f[n-n_0, m-m_0] \xrightarrow{S} g[n-n_0, m-m_0]$

Correlation filtering

- Size of the averaging window: $(2k+1) \times (2k+1)$

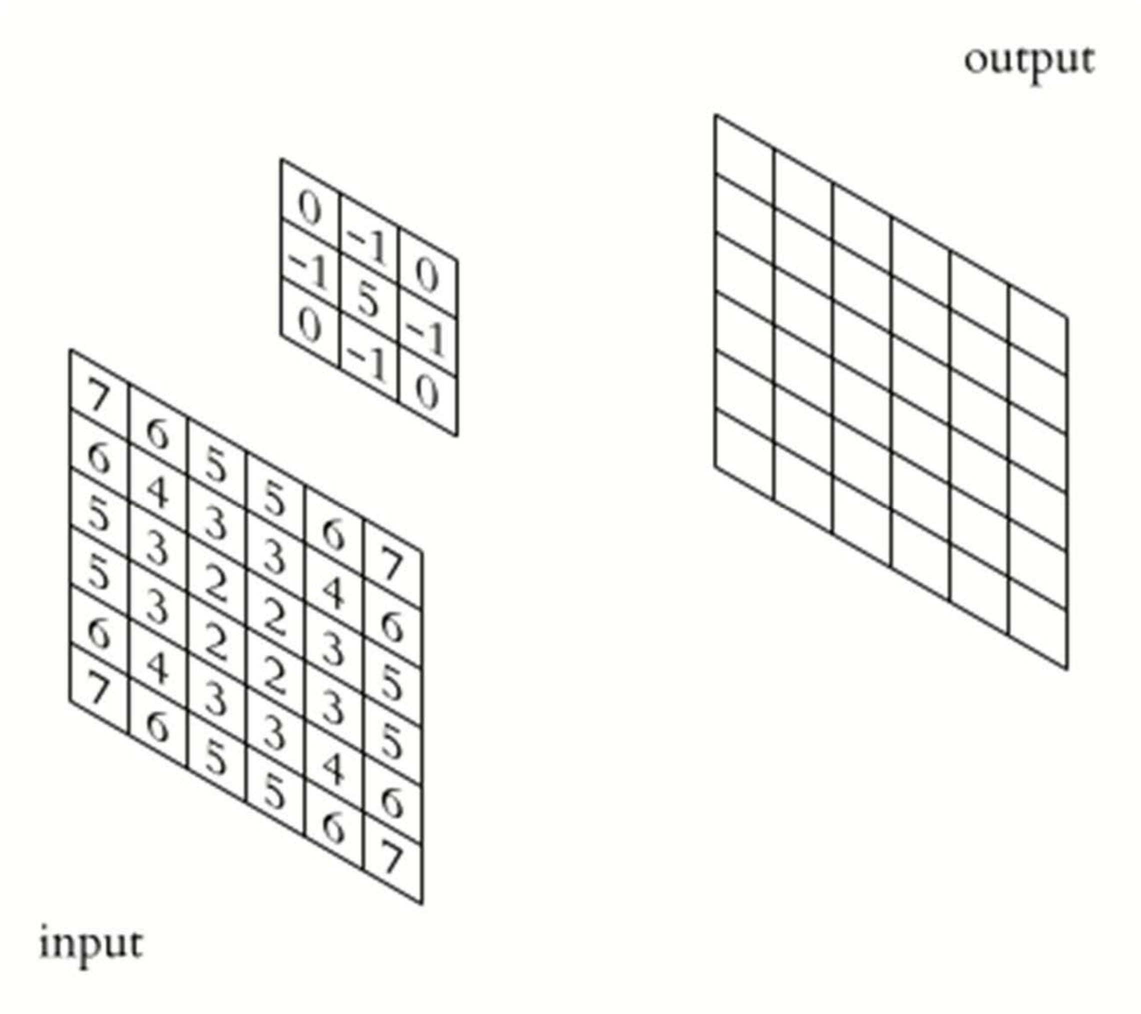
$$g[i,j] = \frac{1}{(2k+1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} f[i+u, j+v]$$

- Generalize to allow different weights depending on neighboring pixel's relative position

$$g[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] f[i+u, j+v]$$

$$g = h \otimes f$$

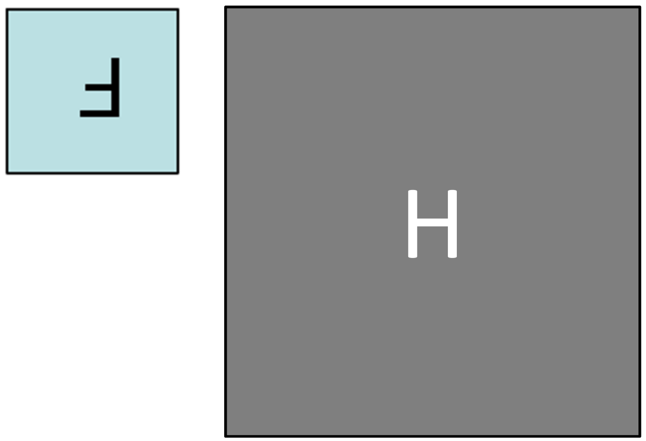
Correlation filtering



Convolution filtering

- Convolution: $g = h \star f$
  - Flip the filter in both dimensions (bottom to top, right to left)
  - Then apply cross-correlation

$$g[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] f[i-u, j-v]$$



Convolution vs. correlation

- Convolution:

$$g[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] f[i-u, j-v]$$

$$g = h \star f$$

- Cross-correlation:

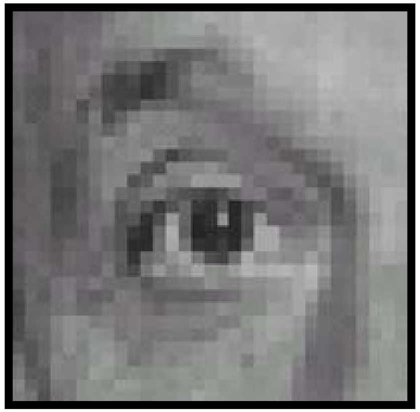
$$g[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h[u,v] f[i+u, j+v]$$

Convolution vs. correlation

- Convolution:
  - Integral that expresses the amount of overlap of one function as it is shifted over another function
  - Convolution is a filtering operation

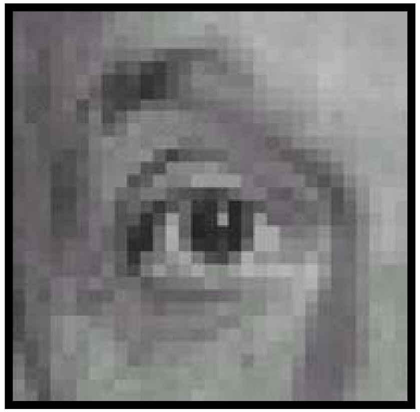
- Cross-correlation:
  - Computes a measure of similarity of two input signals as they are shifted by one another
  - The correlation result reaches a maximum at the time when the two signals match best
  - Correlation is a measure of relatedness of two signals

Linear filter examples



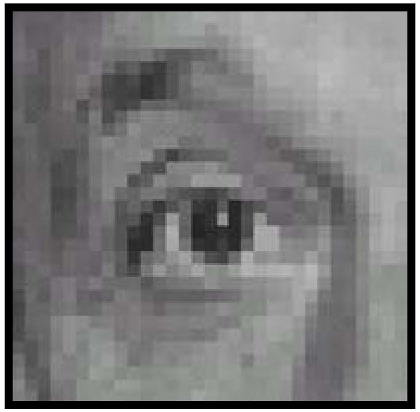
Original

0	0	0
0	1	0
0	0	0



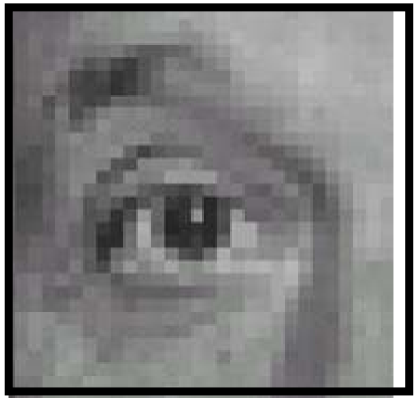
Filtered
(no change)

Linear filter examples



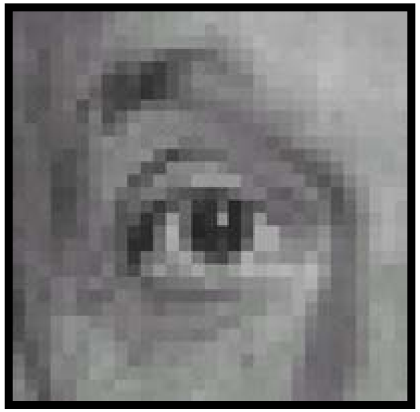
Original

0	0	0
0	0	1
0	0	0



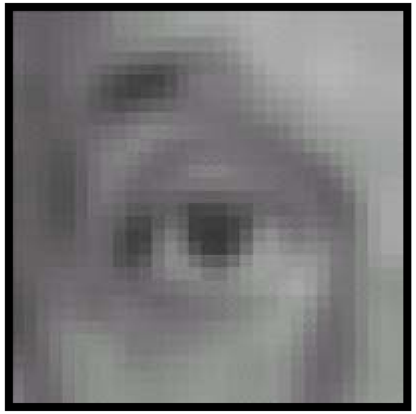
Shifted left
by 1 pixel with
correlation

Linear filter examples



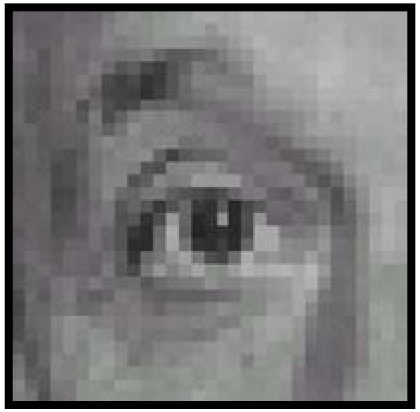
Original

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array}$$



Blur (with a box filter)

Linear filter examples



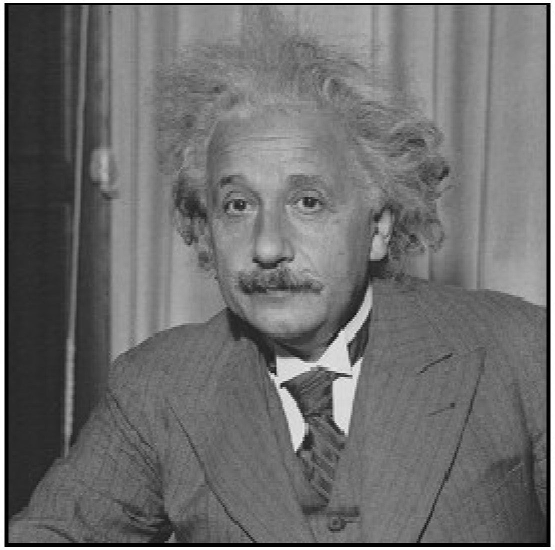
Original

$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

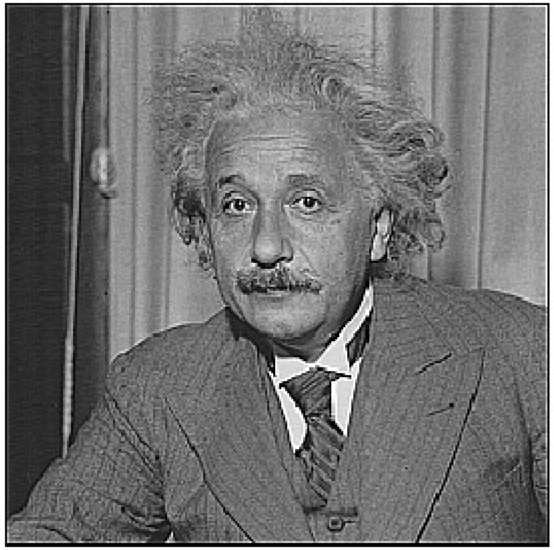
Sharpening filter

Linear filter examples

Sharpening filter



before

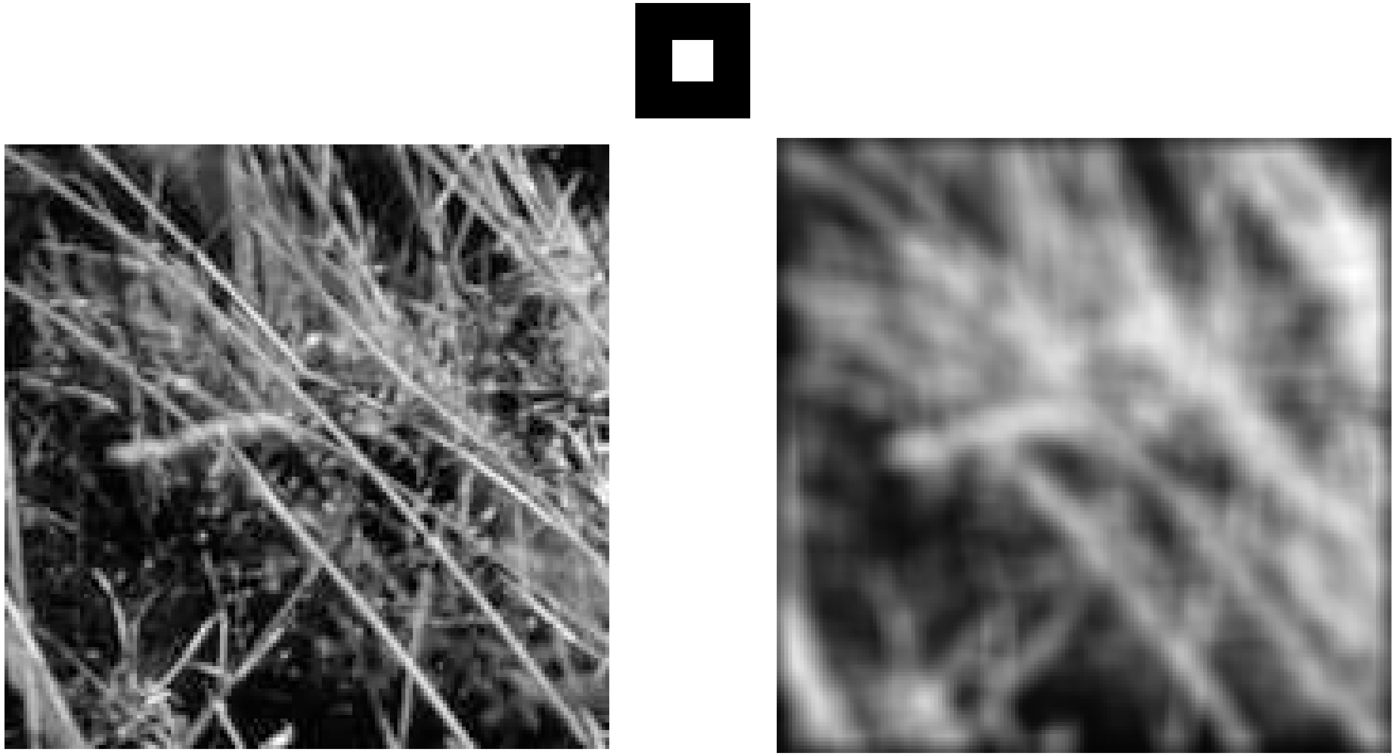


after

Smoothing: Average filter

- We can ***reduce noise by smoothing***

- Smoothing by averaging



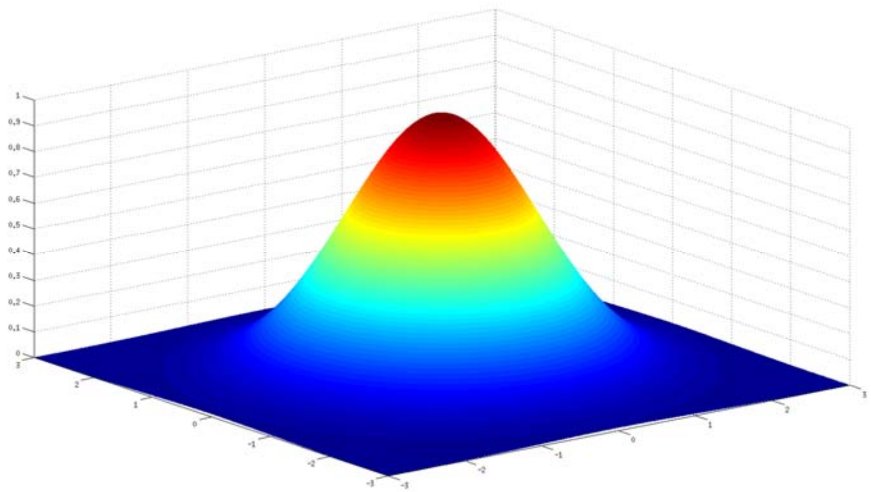
Smoothing: Gaussian filter

- Average smoothing is not appropriate for a defocused lens
  - A single point of light viewed in a defocused lens looks like a fuzzy blob
  - Averaging process would give a little square

- Gaussian kernel gives a good model of a fuzzy blob

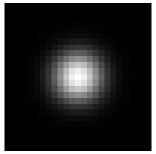
$$h(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$

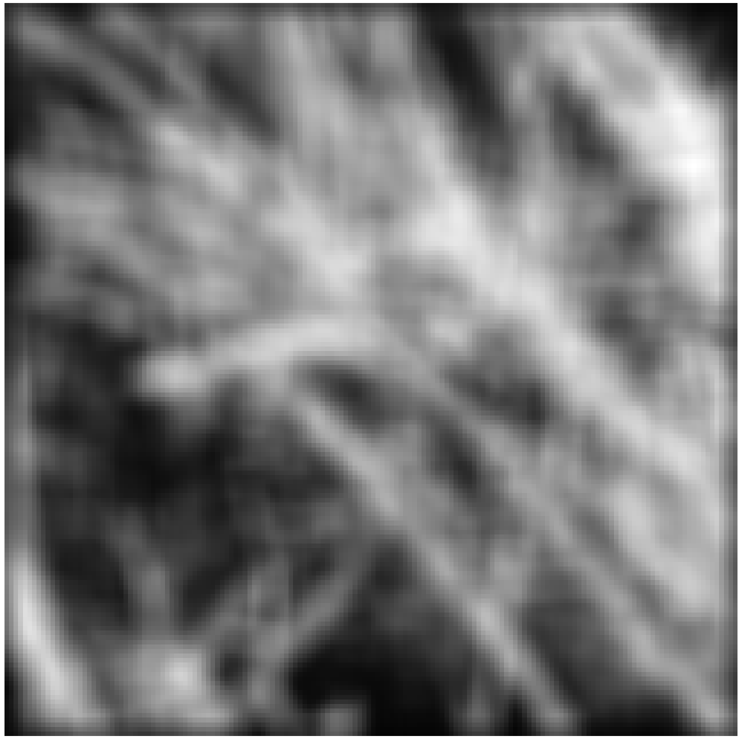
1	2	1
2	4	2
1	2	1



Smoothing: Gaussian filter

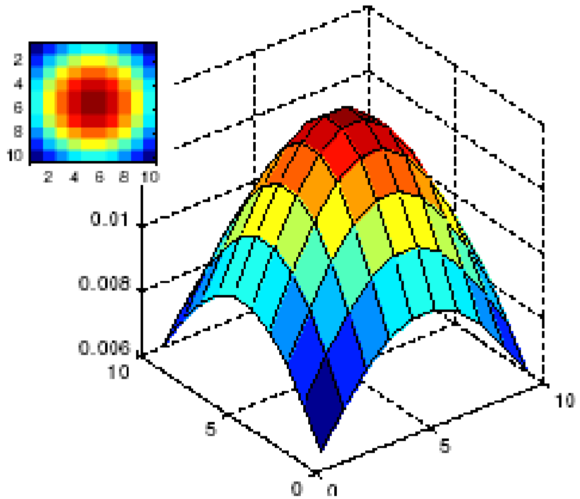
- Smoothing with a Gaussian

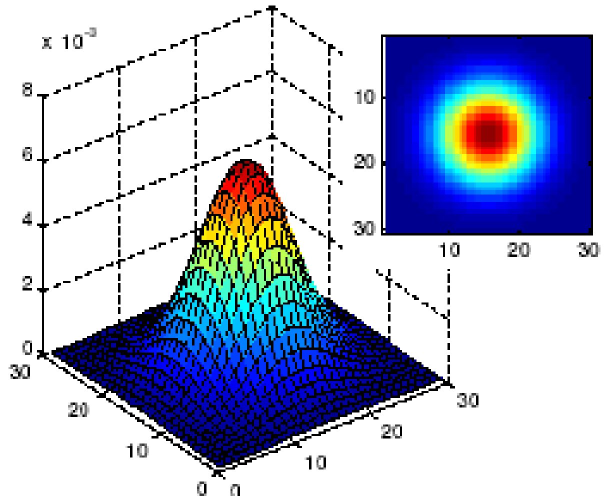




Smoothing: Gaussian filter

- Size of kernel

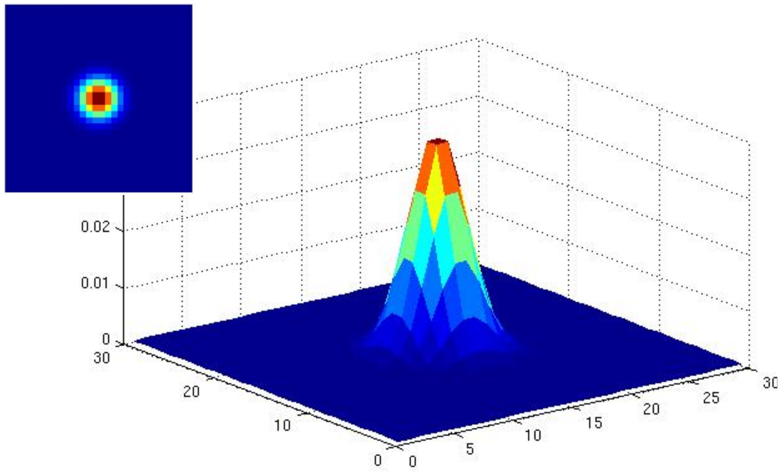




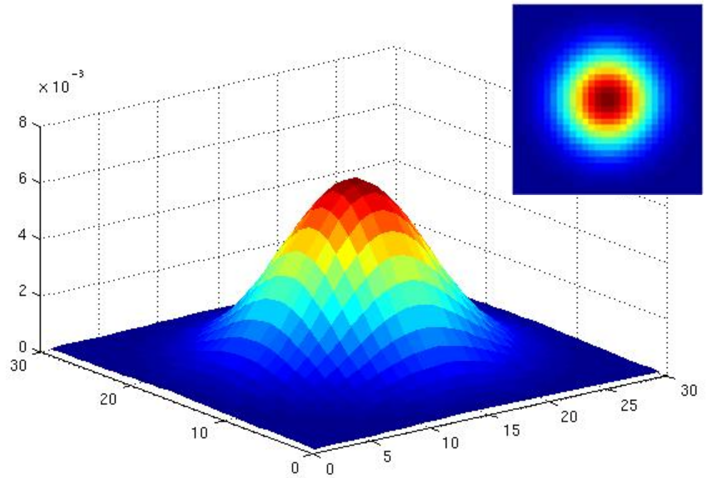
$\boldsymbol{\sigma} = \mathbf{5}$ **with** $\mathbf{10 \times 10}$ **kernel** $\quad$ $\boldsymbol{\sigma} = \mathbf{5}$ **with** $\mathbf{30 \times 30}$ **kernel**

Smoothing: Gaussian filter

- Variance of Gaussian
  - Determines extent of smoothing



$\sigma = 2$ **with** 30×30 **kernel**

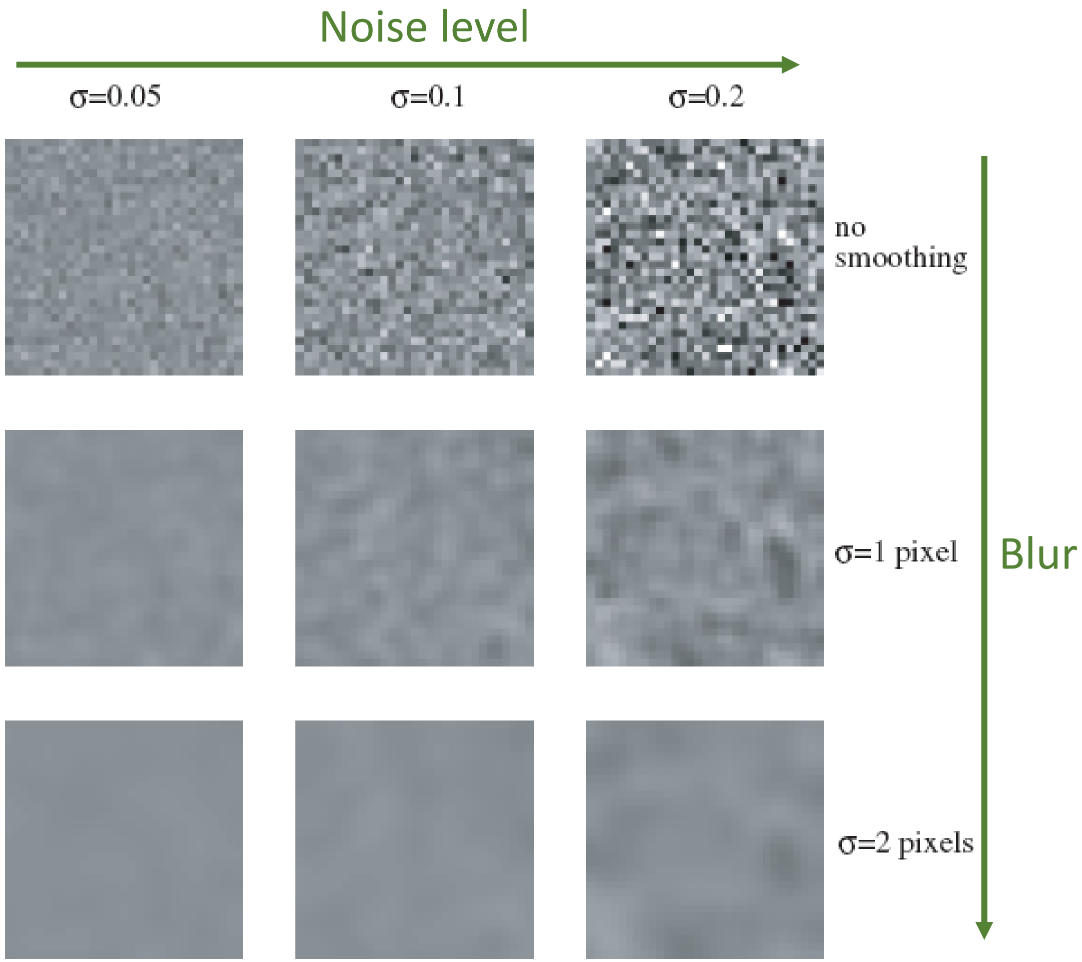


$\sigma = 5$ **with** 30×30 **kernel**

Smoothing: Gaussian filter

- Rows
  - Smoothing with Gaussians of different width

- Columns
  - Different realizations of an image of Gaussian noise

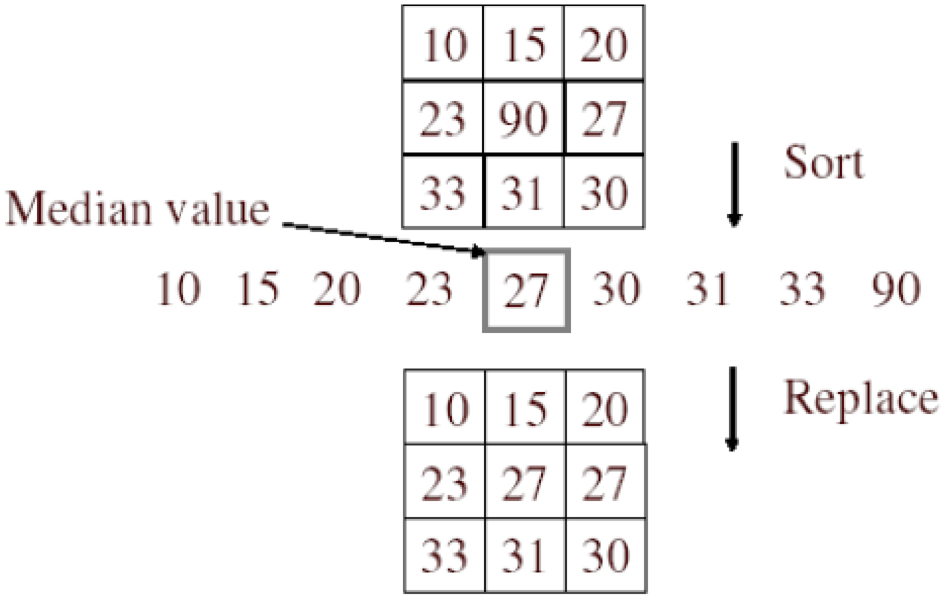


Mean filter

- Smoothing by averaging
  - Good when the average is taken over a homogeneous neighborhood with zero-mean noise
  - When the neighborhood straddles the boundary between two homogeneous regions, the estimate results in blurring of the boundary

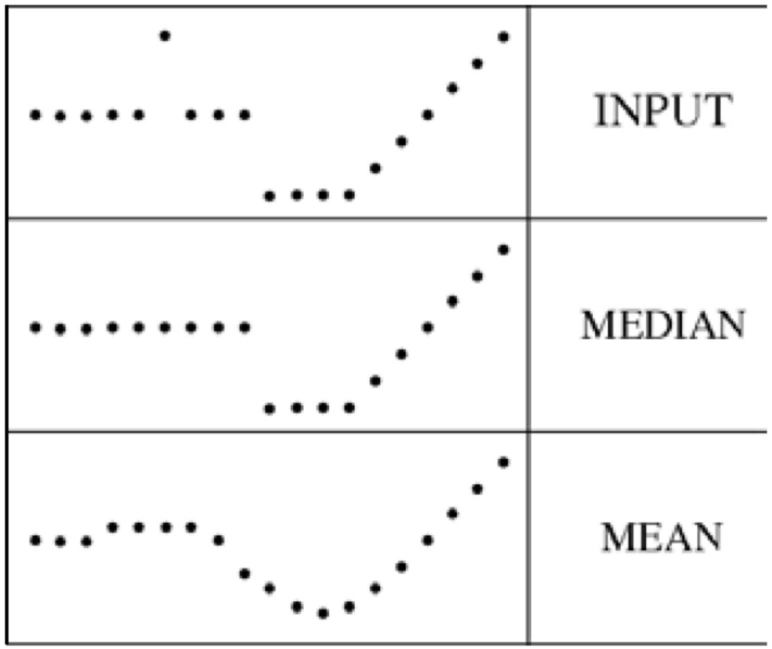
Median filter

- Median filter
  - Non-linear filter
  - Method:
    - 1) Rank-order neighborhood intensities
    - 2) Take middle value
  - No new gray level emerge

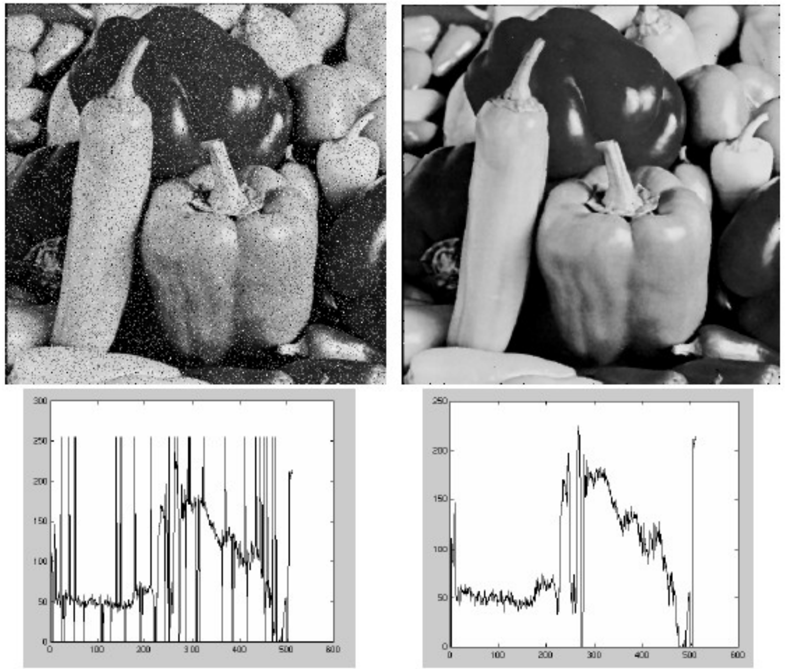


Median filter

- Median filter
  - Remove spikes
    - Good for impulse, salt & pepper noise
  - Less sensitive to outliers compared to mean filter



Salt and pepper noise Median filtered



Plots of a row of the image

Median filter

- Median filter
  - Not always optimistic

3 x 3 median filter

Sharpens edges, destroys edge cusps and protrusions

Median filter

- Median filter
  - Not always optimistic

Comparison with Gaussian filter

E.g.) Upper lip smoother, eye better preserved

Median filter

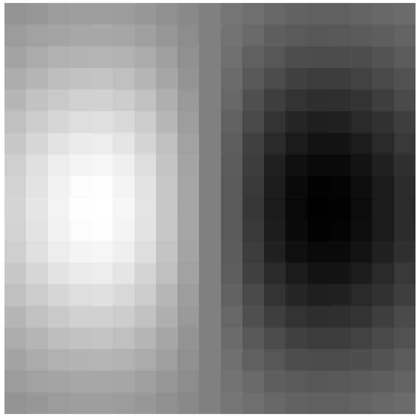
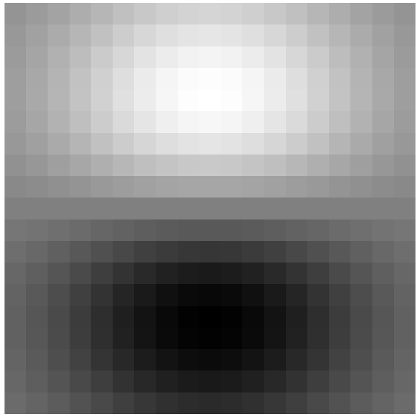
- Median filter
  - Not always optimistic

10 times 3 x 3 median filter

Patchy effect: Important details lost (e.g., earring)

Filters as templates

- Filtering the image
  - Applying a filter at some point can be seen as taking a dot-product between the image and some vector
  - Filtering the image is a set of dot products
- Filters look like the effects they are intended to find matched filters

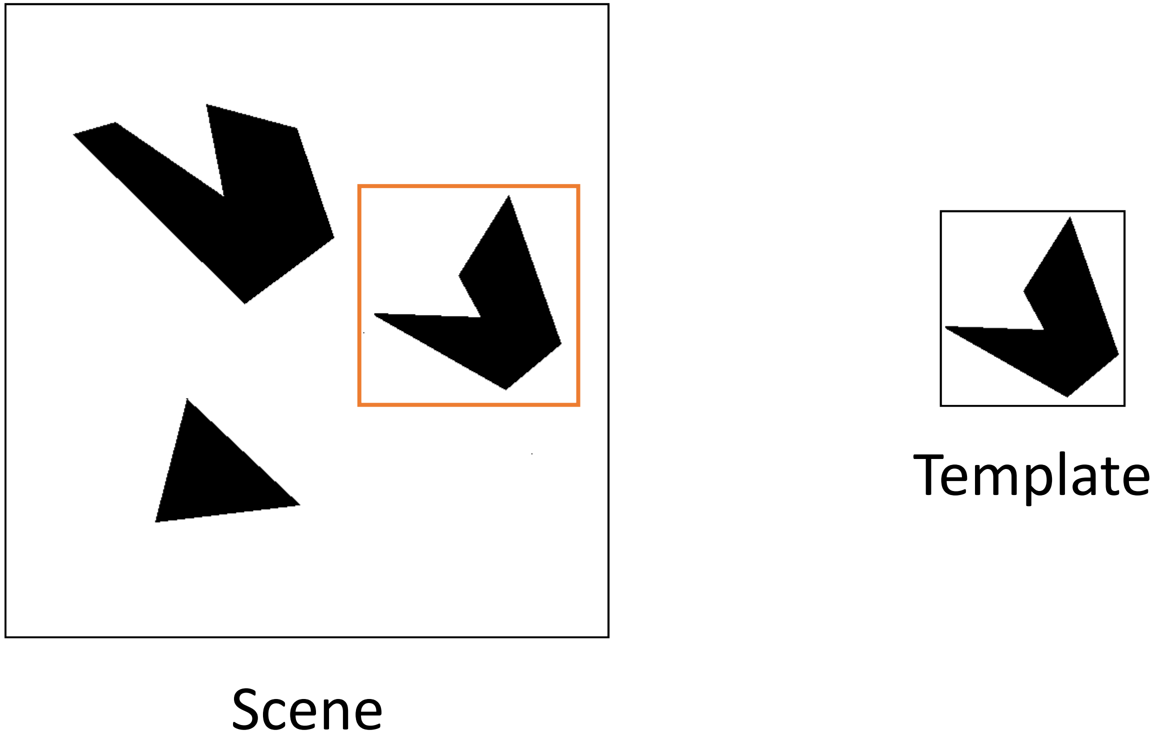



Filters as templates

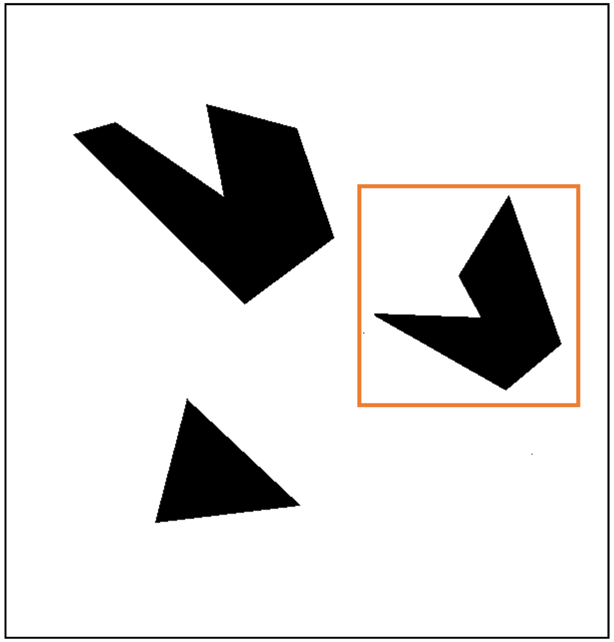
- Filters as templates
  - Can be used for template matching
  - Use normalized cross-correlation to find a given pattern in the image

$$\frac{\sum_{x,y}\left(A_{x,y}-A_{mean}\right)\left(B_{x,y}-B_{mean}\right)}{\sqrt{\sum_{x,y}\left(A_{x,y}-A_{mean}\right)^2}\sqrt{\sum_{x,y}\left(B_{x,y}-B_{mean}\right)^2}}$$

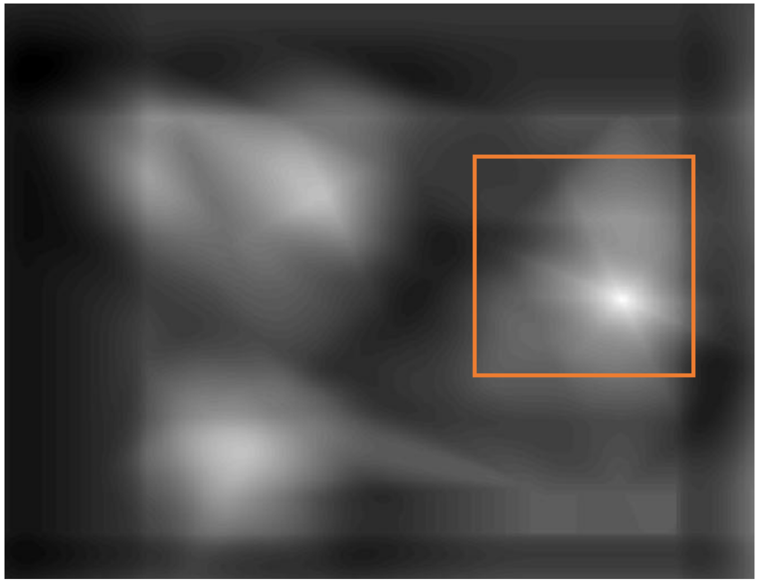
Template matching



Template matching

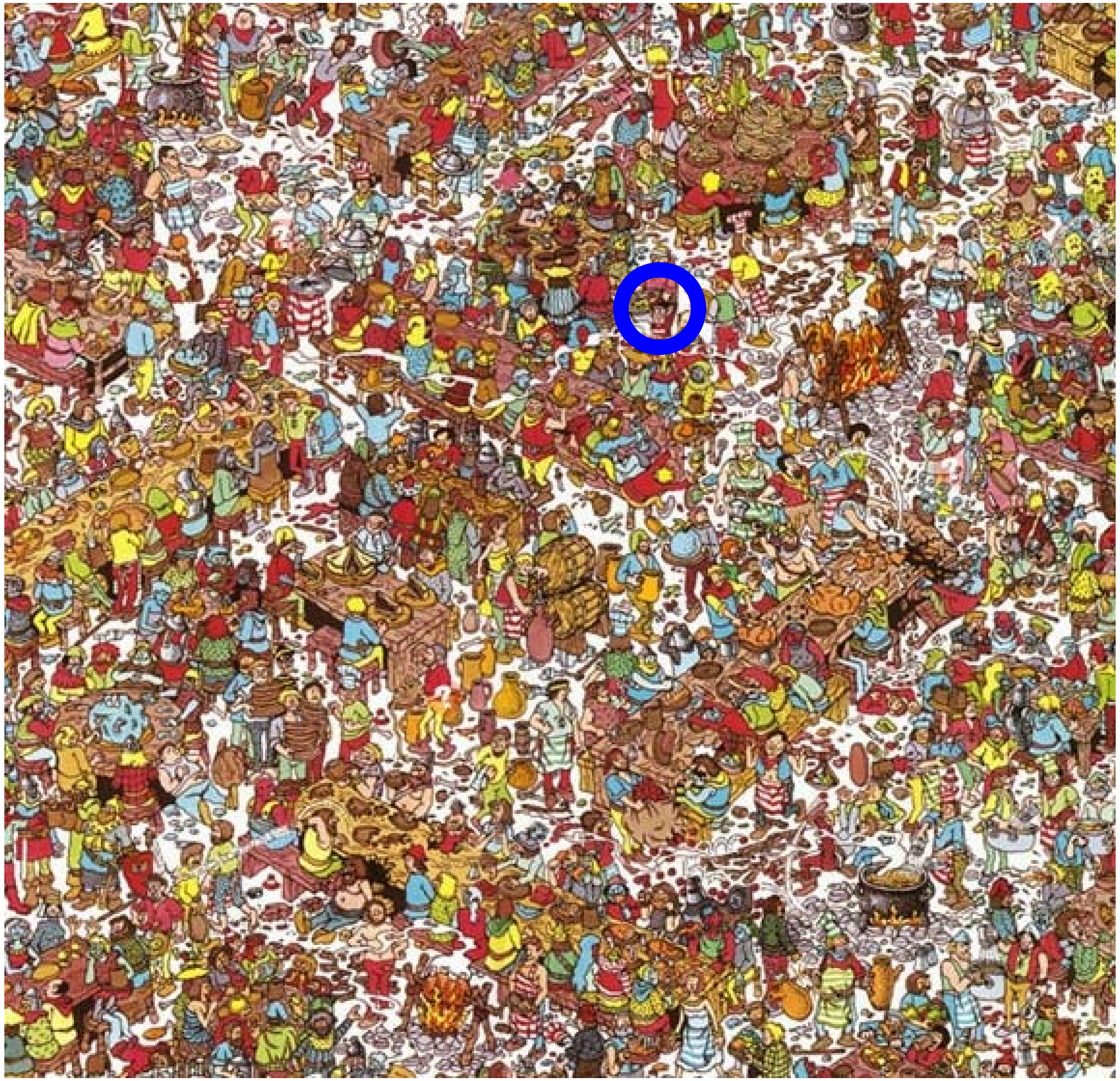


Detected template



Correlation map

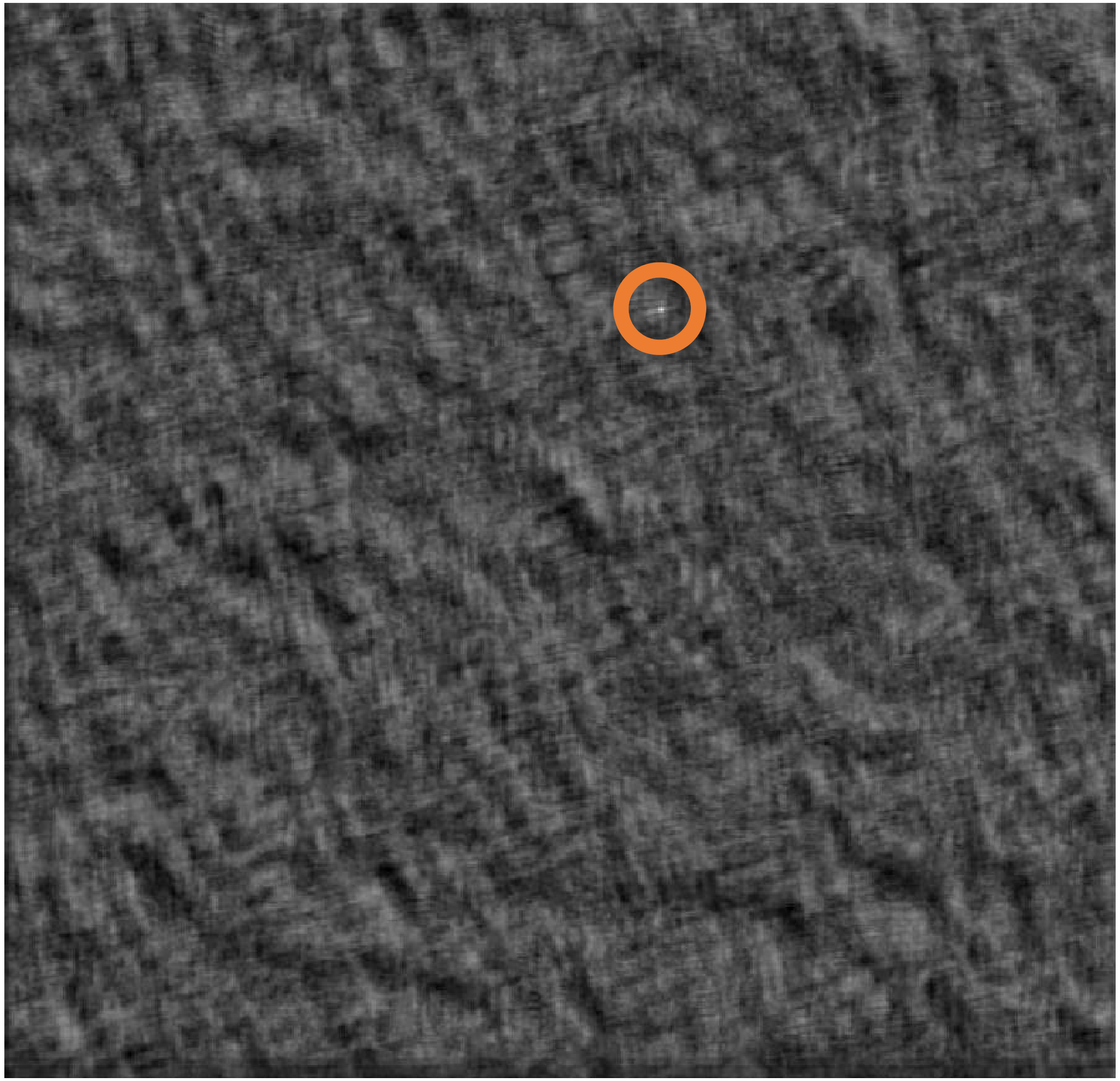
Template matching



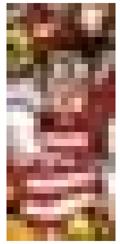
Scene

Template

Template matching



Correlation map



Template