



S A I R

Spatial AI & Robotics Lab

CSE 473/573-A

L5: FILTERING

Chen Wang

Spatial AI & Robotics Lab

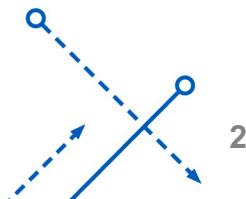
Department of Computer Science and Engineering



University at Buffalo The State University of New York

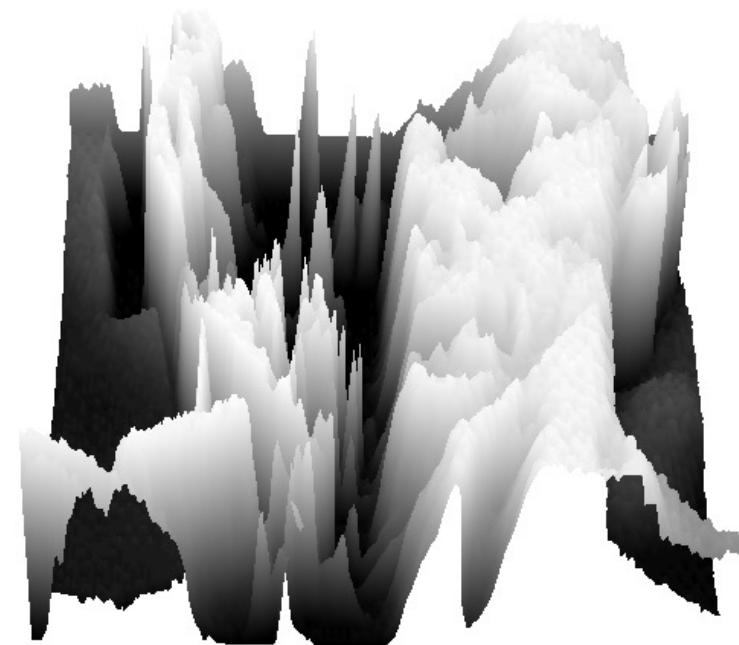
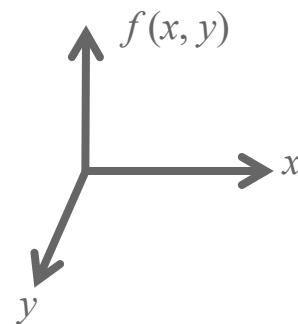
Content

- Filtering
 - Linear filters
 - Correlation and Convolution
 - Equivariance, Invariance
 - Smoothing, Gaussian Filter, Median filter



Recap: Image representation

- A (grayscale) image as a **function**, f , from \mathbb{R}^2 to \mathbb{R} :
 - $f(x, y)$ gives the **intensity** at position (x, y) .
 - A **digital image** is a discrete (**sampled, quantized**) version of this function.



Recap: Images as functions

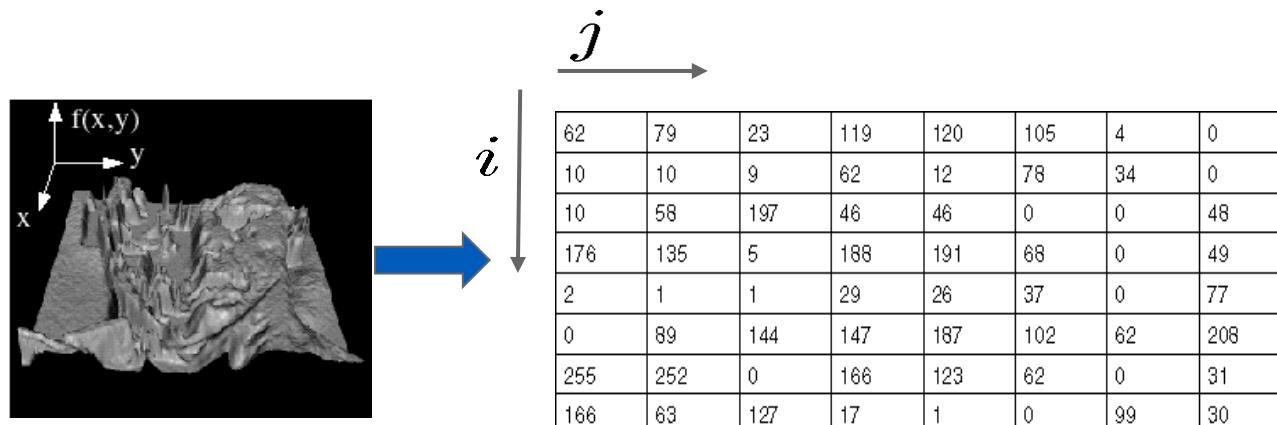
- Take an image as a function, f , from \mathbb{R}^2 to \mathbb{R} :
 - $f(x, y)$ gives the intensity at position (x, y)
 - Realistically, we expect the image only to be defined over a rectangle, with a finite range:
 - $f: [a, b] \times [c, d] \rightarrow [0, 255]$
 - Important: we often convert $[0, 255]$ to **[0,1.0]**.
- A color image is three functions pasted together, a “vector-valued” function

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

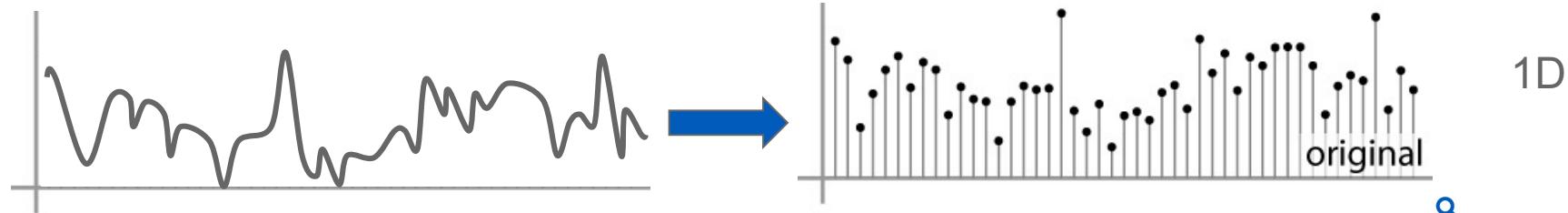


Recap: Digital images

- In computer vision, we operate on **digital (discrete)** images:
 - Sample the 2D space on a regular grid
 - Quantize each sample (round to nearest integer)
- Image thus represented as a matrix of integer values.



2D



1D

Adapted from S. Seitz

Recap: Warping v.s. Filtering

image warping: change **domain** of image

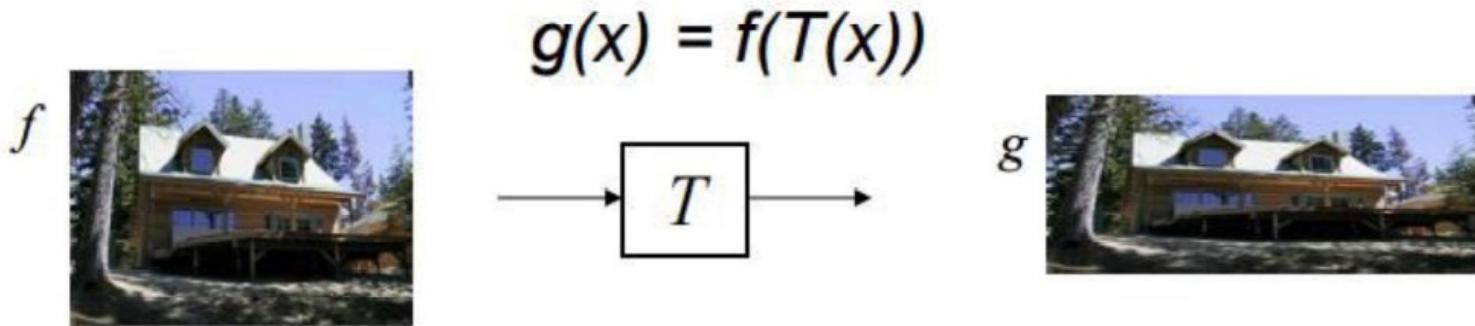


image filtering: change **range** of image (Next Week)

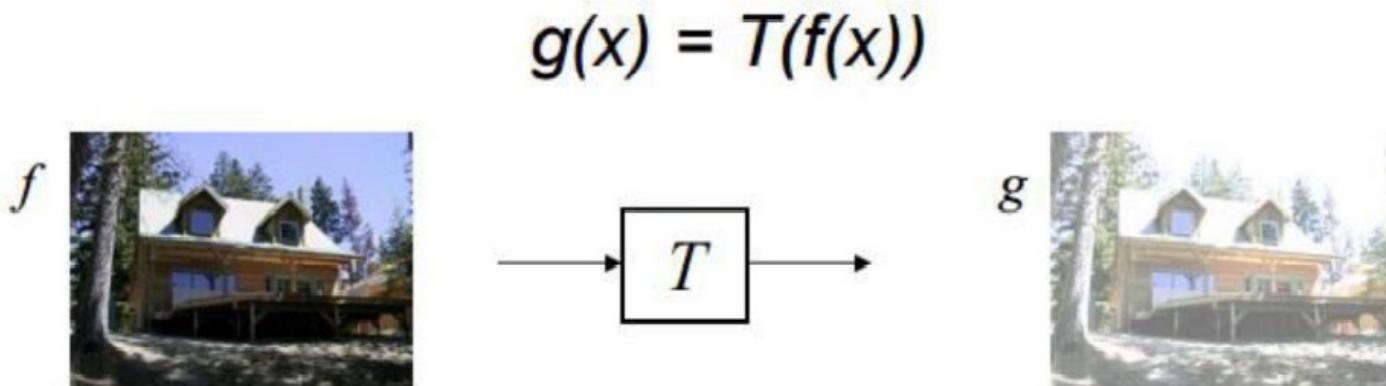


Image filtering

- Image filtering: compute a function of the local neighborhood at each position
- Really important!
 - Enhance images
 - Denoising, resize, increase contrast, etc.
 - Extract information from images
 - Texture, edges, distinctive points, etc.
 - Detect patterns
 - Template matching
 - Deep Convolutional Networks

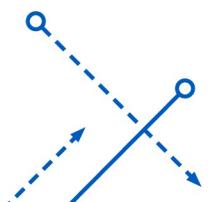
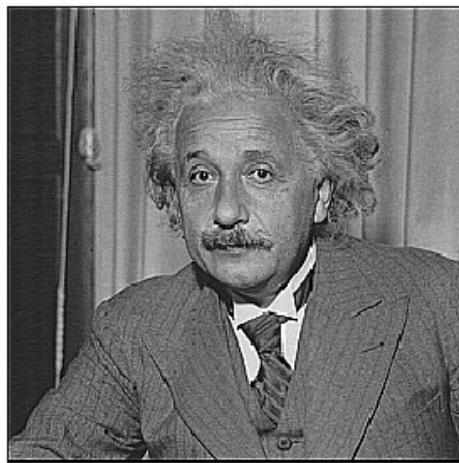
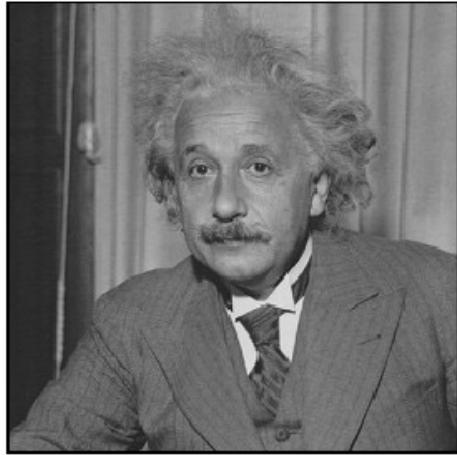


Image Filtering



Smooth/Sharpen Images...



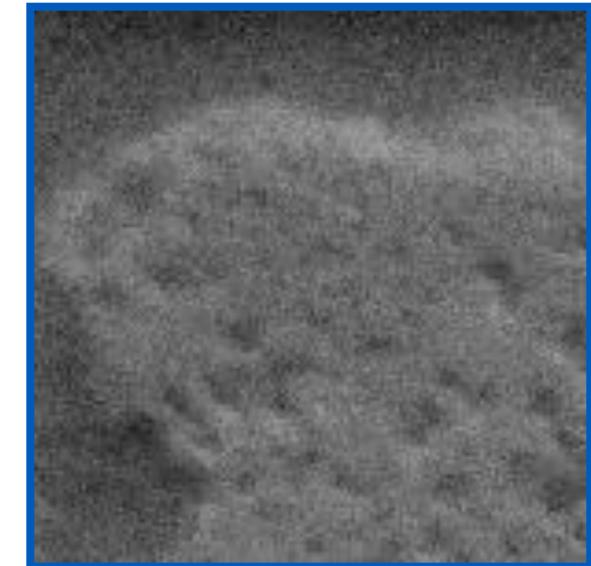
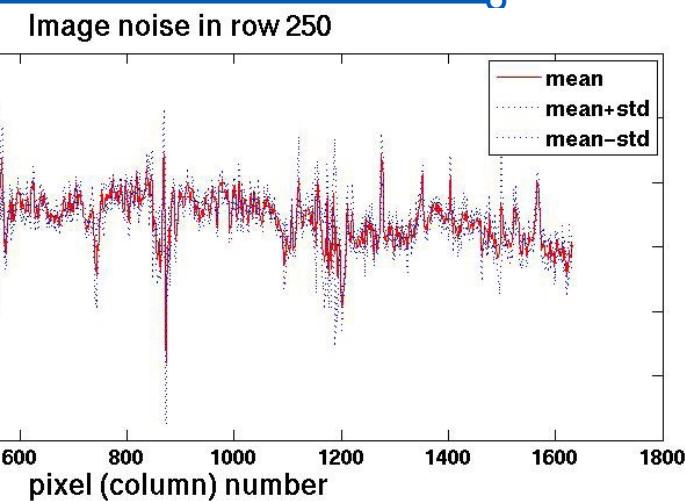
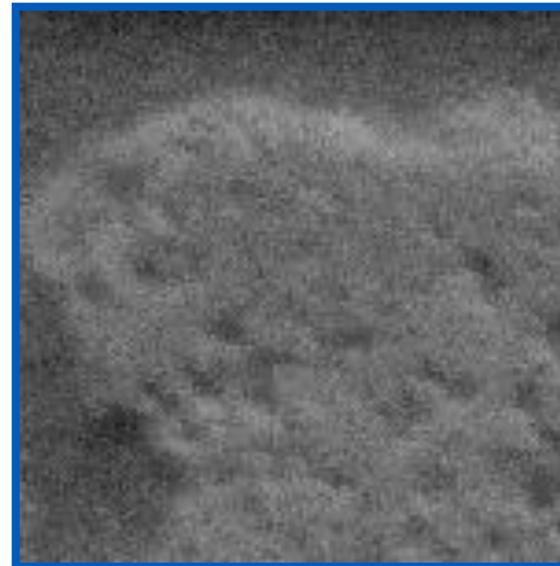
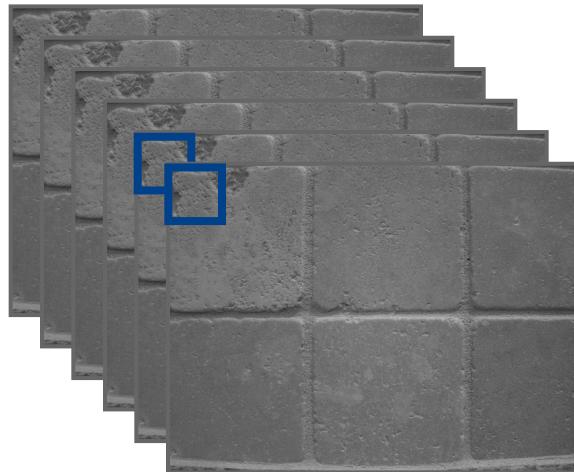
Find edges...



Find Waldo...

How can we do noise reduction?

- We can measure **noise** in multiple images of the same static scene.
- How could we reduce the noise, i.e., give an estimate of the true intensities?



Common types of noise

- **Impulse noise:** random occurrences of white pixels
- **Salt and pepper noise:** random occurrences of black and white pixels
- **Gaussian noise:** variations in intensity drawn from a Gaussian normal distribution



Original



Salt and pepper noise



Impulse noise



Gaussian noise

Gaussian noise

- Additive Noise

$$f(x, y) = \overbrace{\hat{f}(x, y)}^{\text{Ideal Image}} + \overbrace{\eta(x, y)}^{\text{Noise process}}$$

Gaussian i.i.d. ("white") noise:
 $\eta(x, y) \sim \mathcal{N}(\mu, \sigma)$

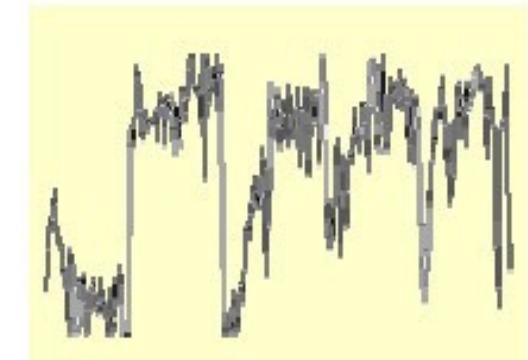
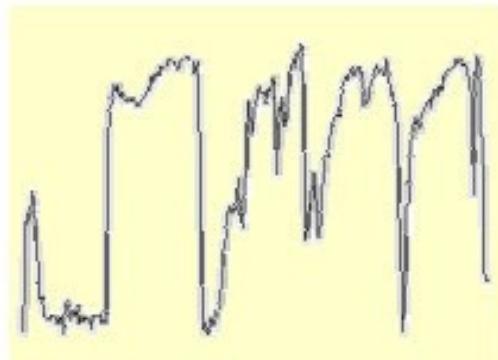
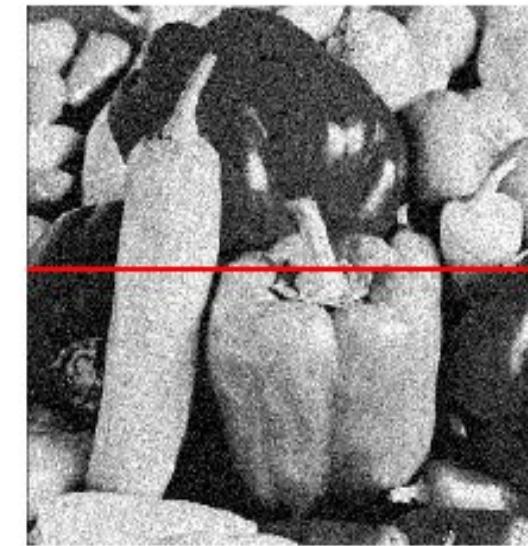
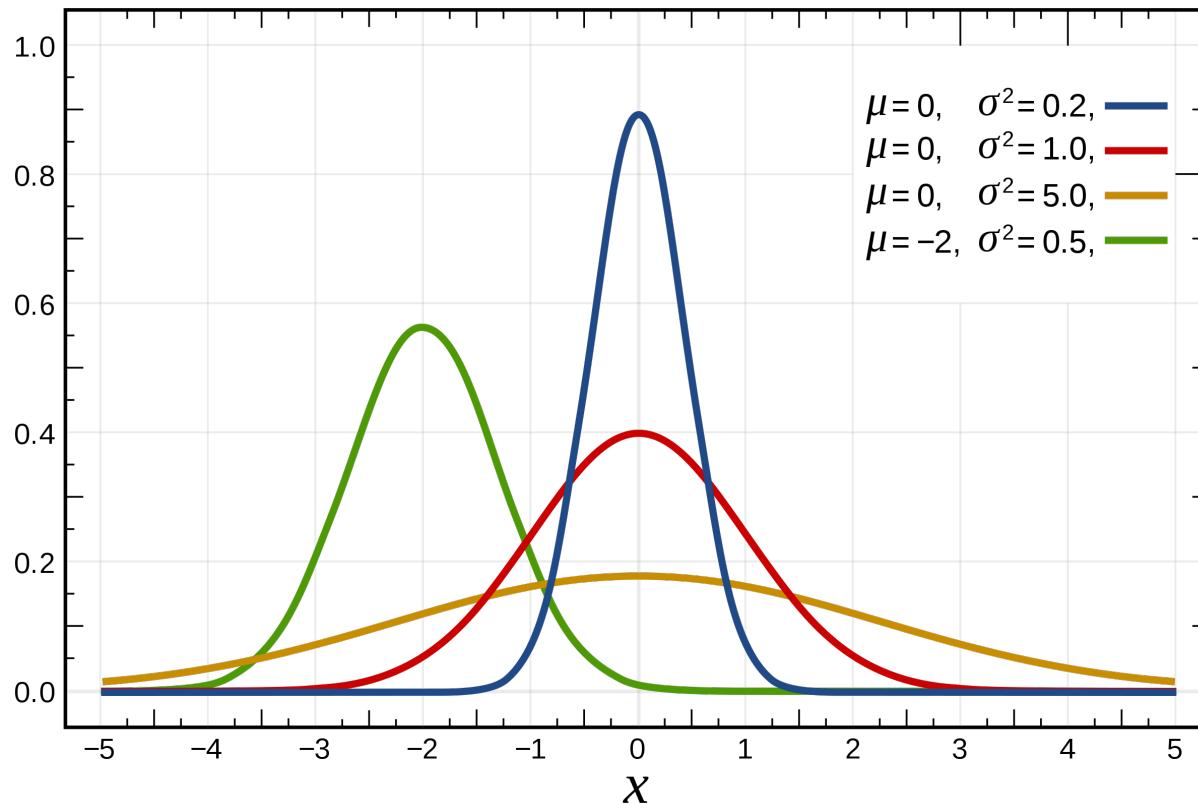


Fig: M. Hebert

PDF of Gaussian distribution

- Probability density function



$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Gaussian noise

$\sigma = 1$

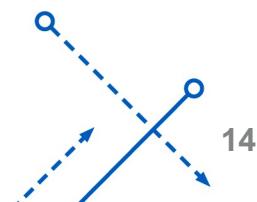
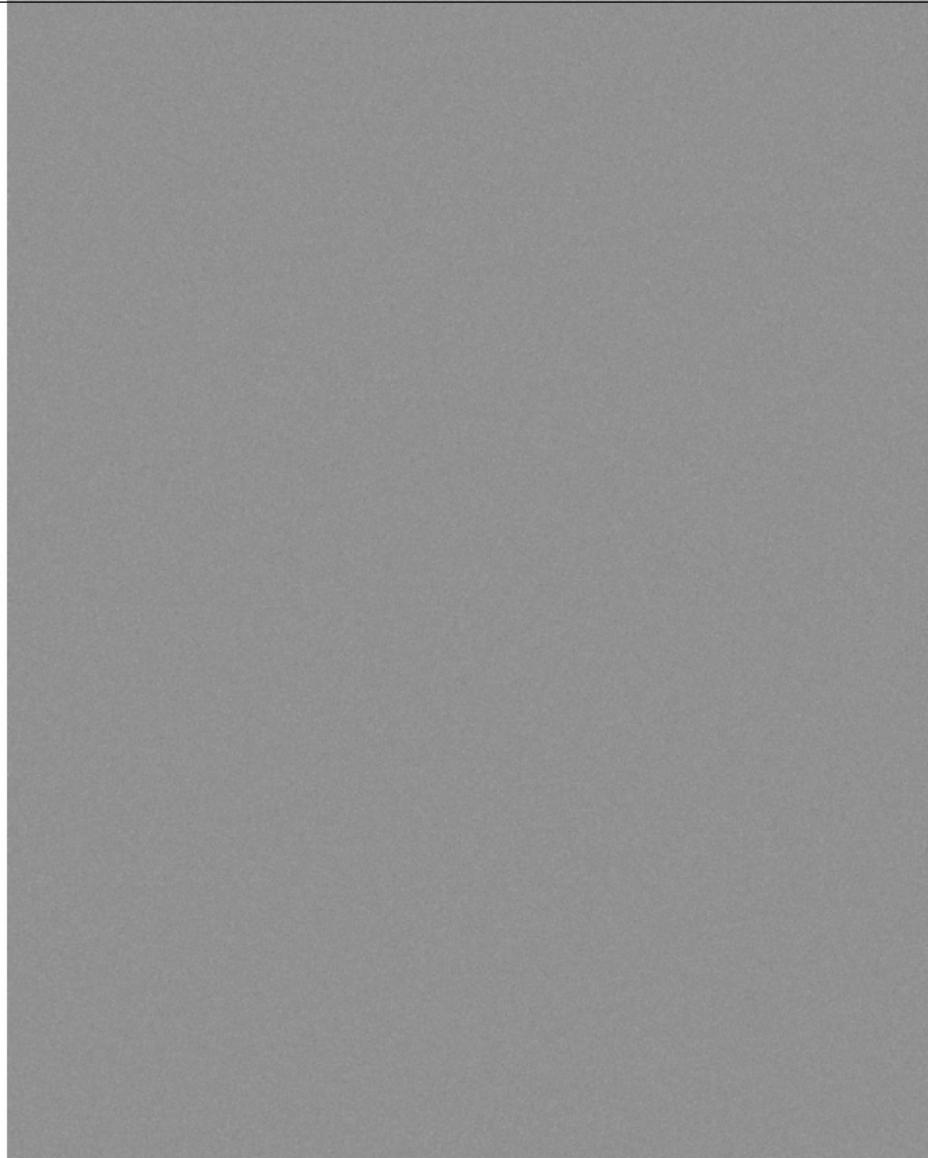


Effect of σ on Gaussian noise:

Image shows the noise values themselves.

Gaussian noise

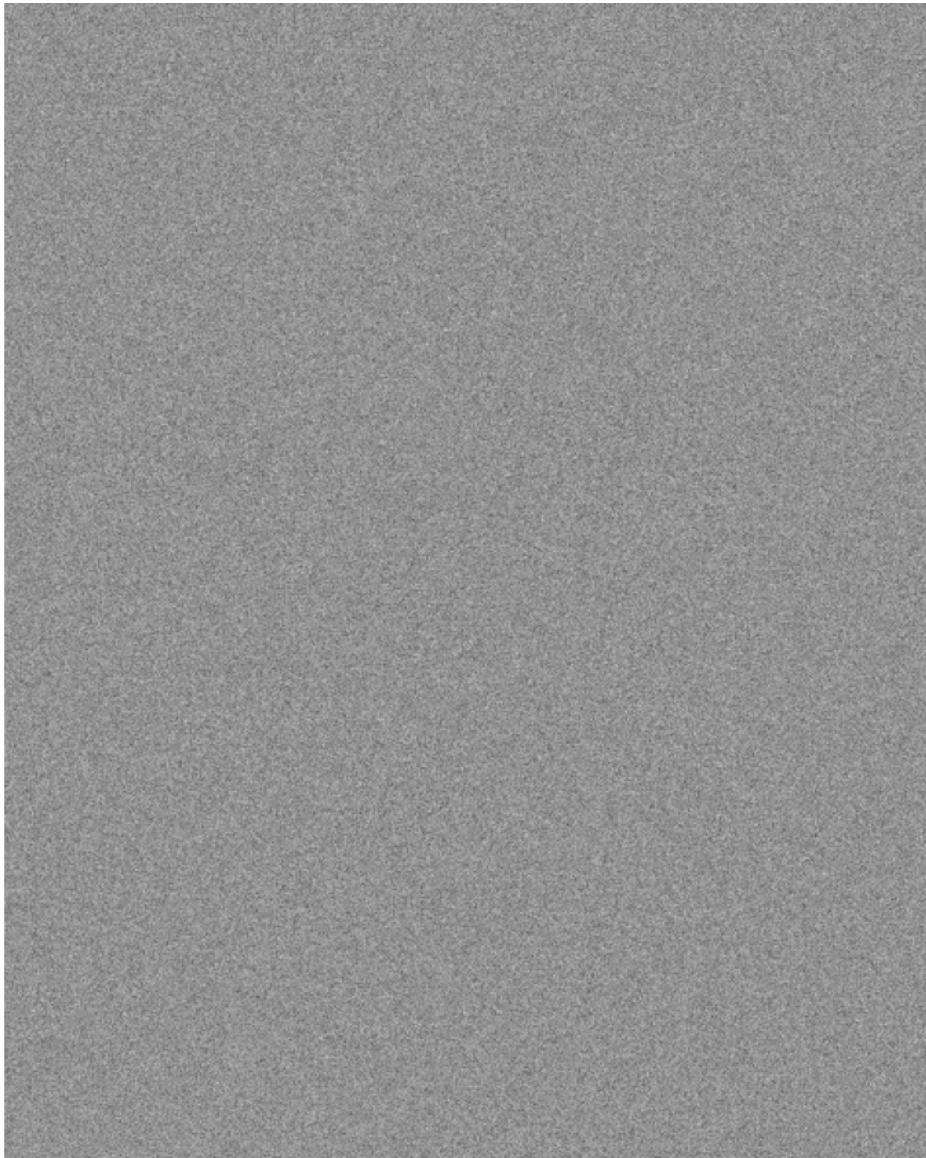
$\sigma = 4$



14

Gaussian noise

$\sigma = 16$



15

Gaussian noise

$\sigma = 1$



Effect of sigma on Gaussian noise:

This shows the noise values added to the raw intensities of an image.

Gaussian noise

$\sigma = 16$



Pixel neighborhoods are important.

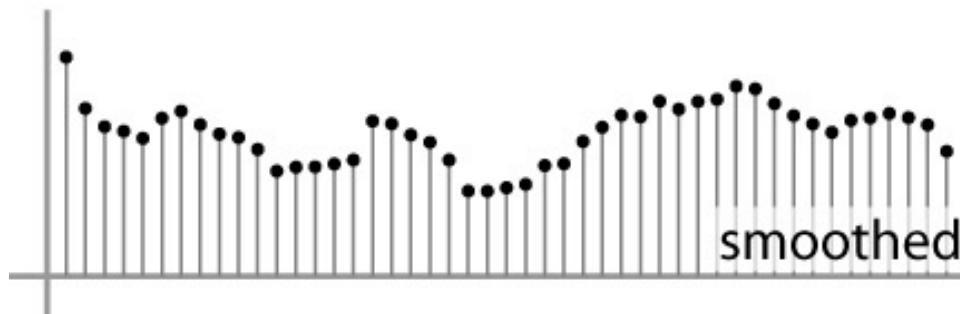
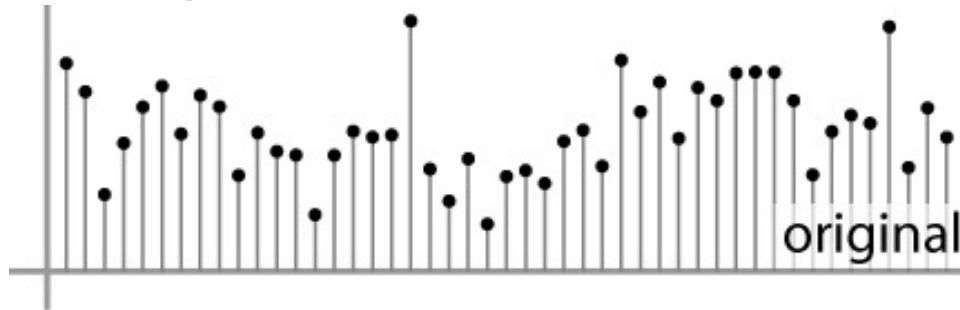
- Q: What happens if we reshuffle all pixels?



- A: Its histogram won't change.
Point-wise processing unaffected.
- Can we use neighborhoods to remove image noise?

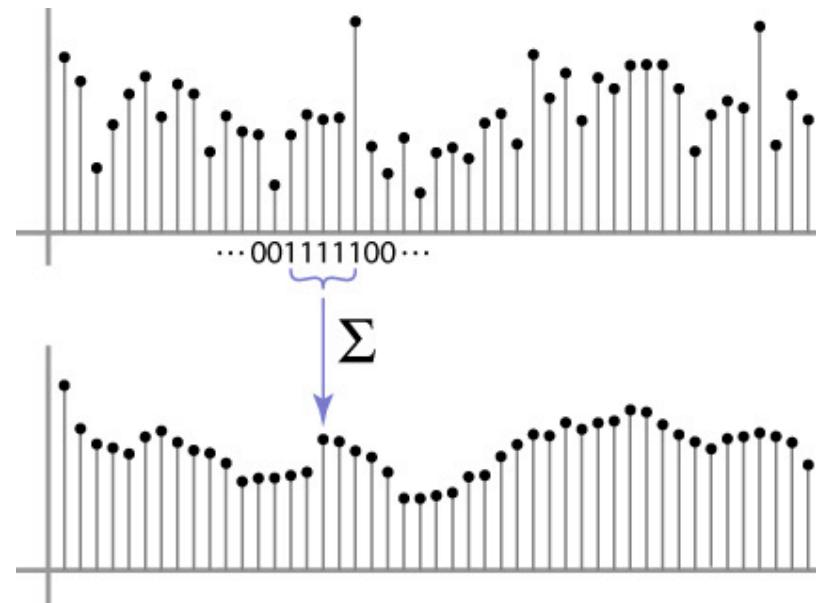
First attempt at a solution

- Let's replace each pixel with an average of all the values in its neighborhood
 - Expect pixels to be like their neighbors
 - Expect noise processes to be independent from pixel to pixel.
 - Moving average in 1D:



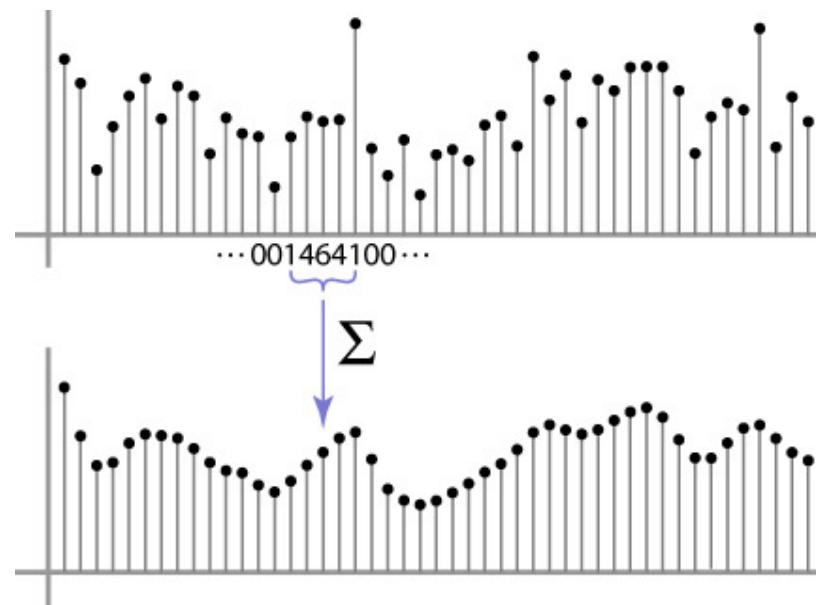
Weighted Moving Average

- We can add weights to moving average
- $Weights [1, 1, 1, 1, 1] / 5$



Weighted Moving Average

- Non-uniform weights $[1, 4, 6, 4, 1] / 16$



Example: Box Filter

$$f[\cdot, \cdot]$$

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

$$h[\cdot, \cdot]$$

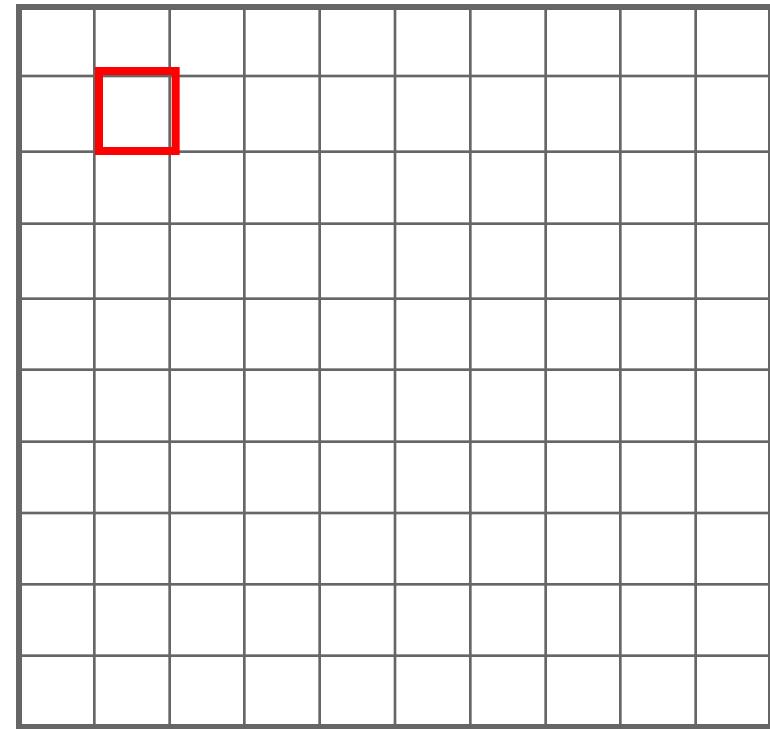
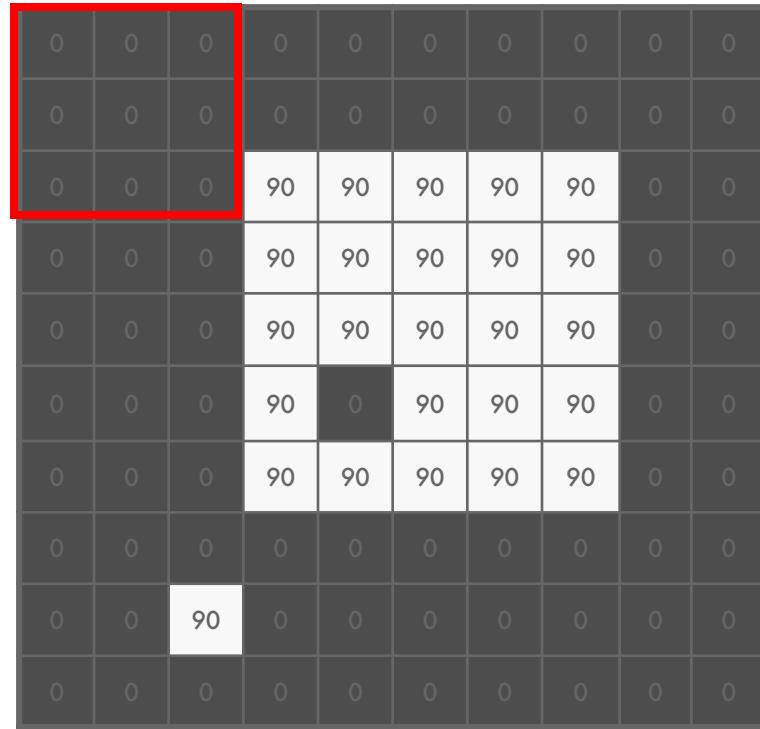
$$\frac{1}{9}$$

1	1	1
1	1	1
1	1	1

Box Filter

 $f[\cdot, \cdot]$ $g[\cdot, \cdot]$ $h[\cdot, \cdot]$

1	1	1
1	1	1
1	1	1



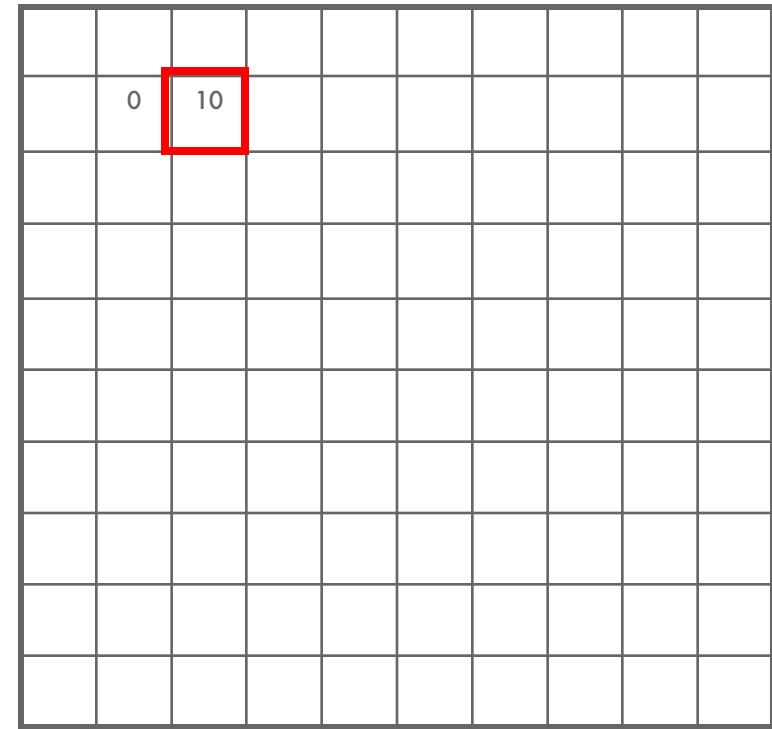
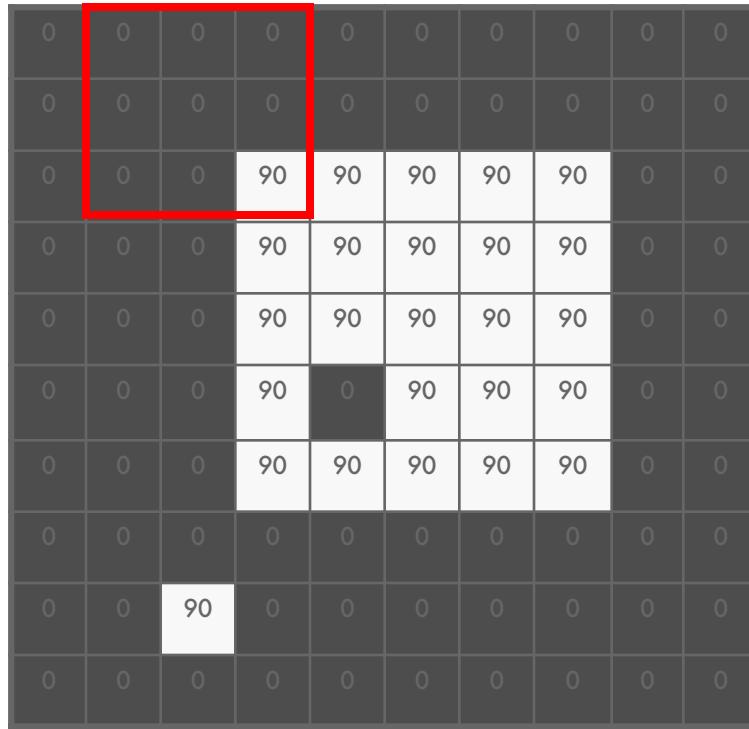
$$g[m, n] = \sum_{k,l} h[k, l] f[m + k, n + l]$$

Credit: S. Seitz
23

Box Filter

 $f[\cdot, \cdot]$ $g[\cdot, \cdot]$ $h[\cdot, \cdot]$

1	1	1
1	1	1
1	1	1

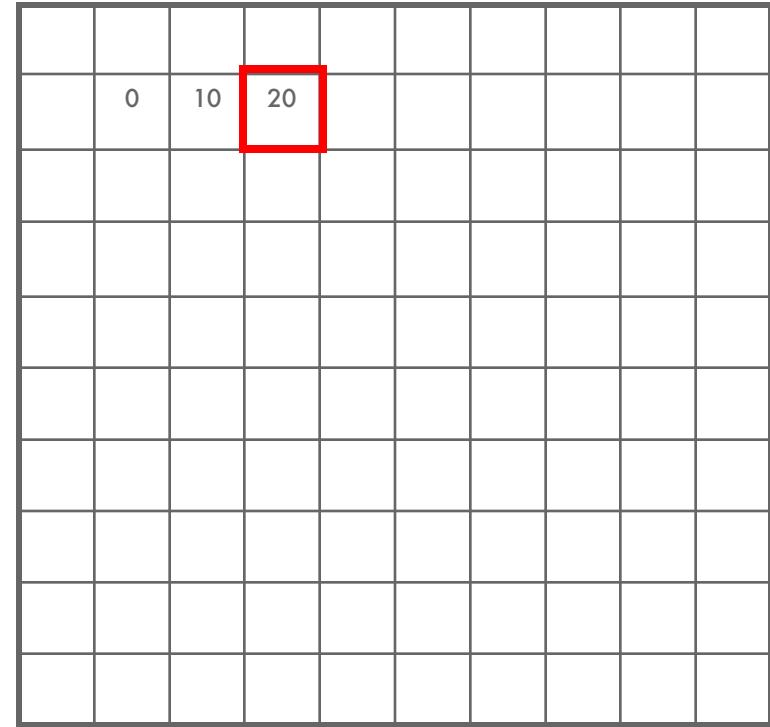
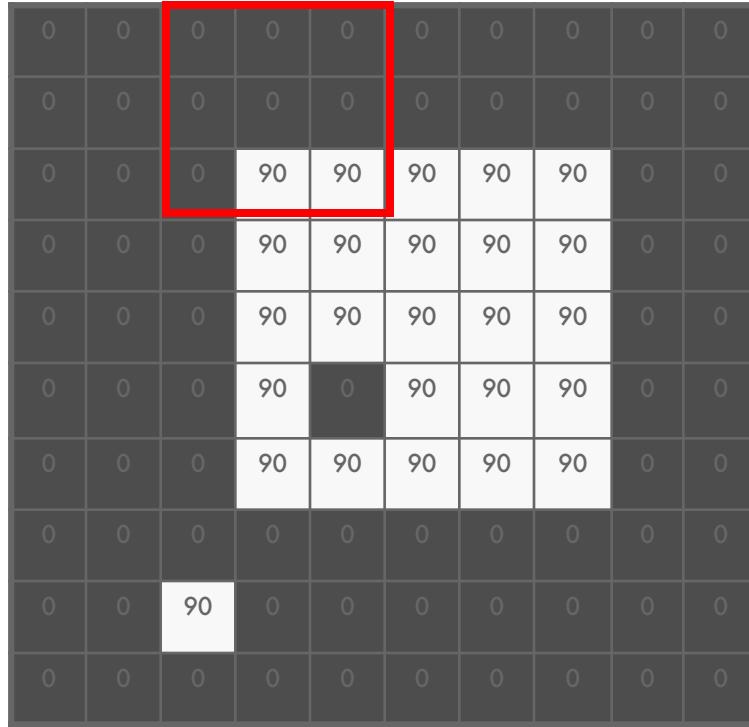


$$g[m, n] = \sum_{k,l} h[k, l] f[m + k, n + l]$$

Box Filter

 $f[\cdot, \cdot]$ $g[\cdot, \cdot]$ $h[\cdot, \cdot]$

1	1	1
1	1	1
1	1	1

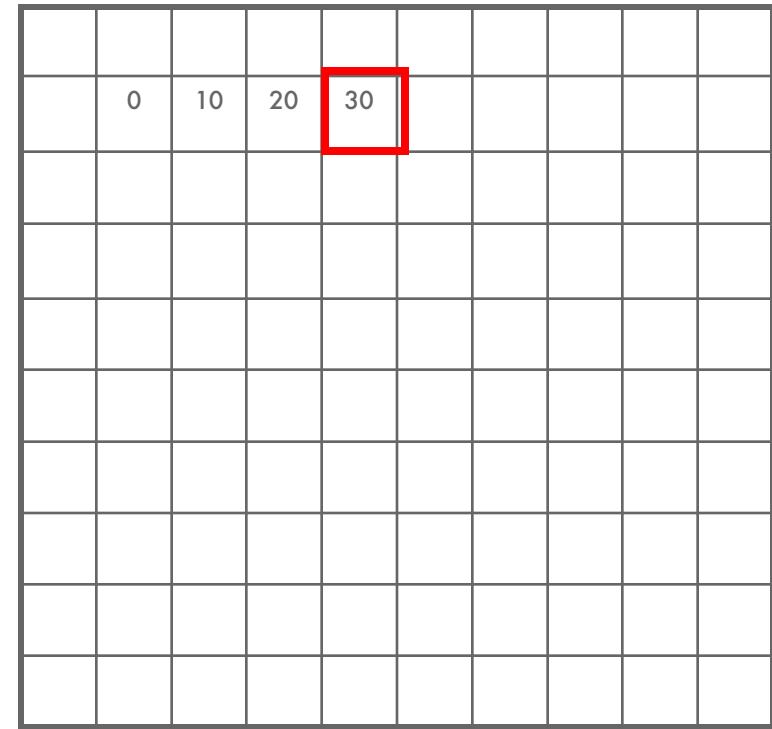
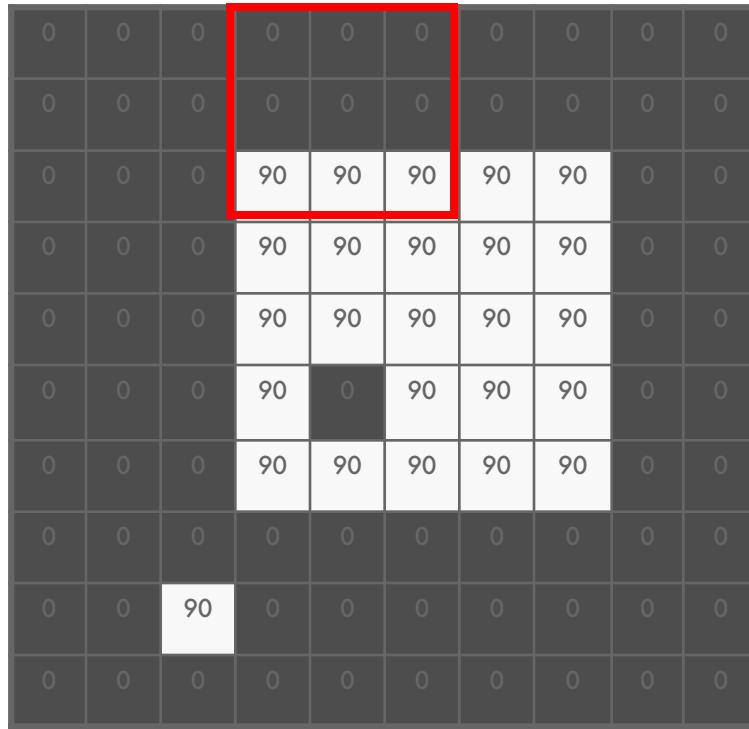


$$g[m, n] = \sum_{k,l} h[k, l] f[m + k, n + l]$$

Box Filter

 $f[\cdot, \cdot]$ $g[\cdot, \cdot]$ $h[\cdot, \cdot]$

1	1	1
1	1	1
1	1	1

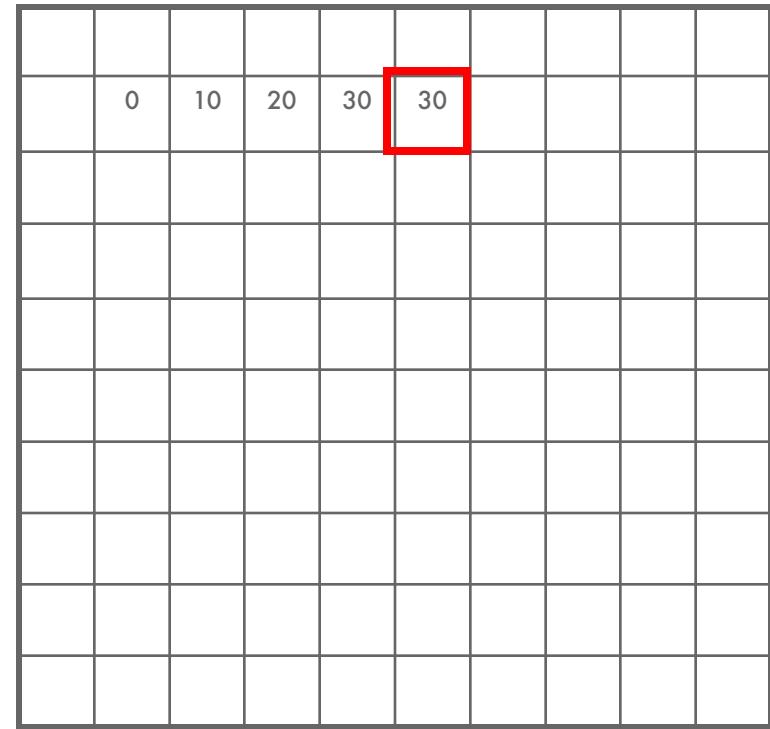
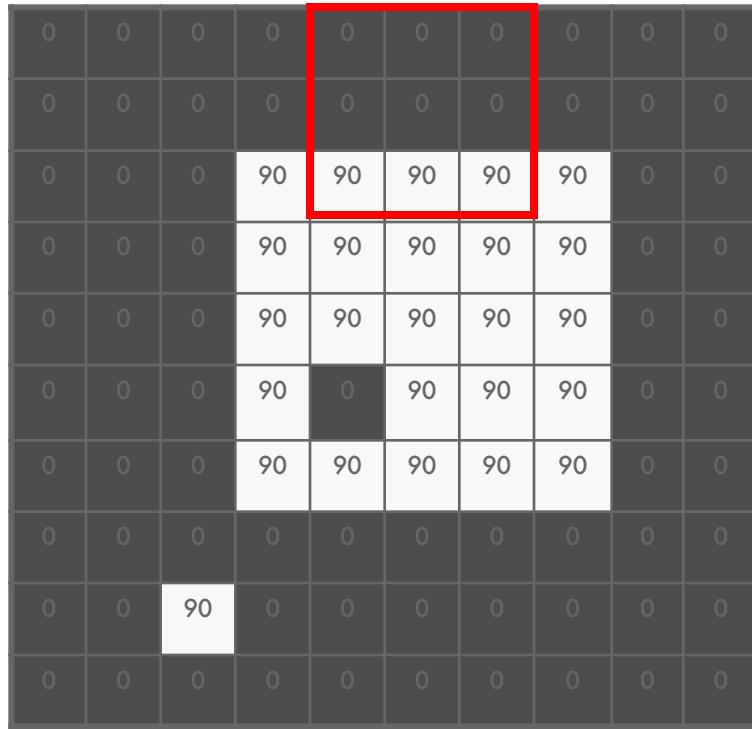


$$g[m, n] = \sum_{k,l} h[k, l] f[m + k, n + l]$$

Box Filter

 $f[\cdot, \cdot]$ $g[\cdot, \cdot]$ $h[\cdot, \cdot]$

1	1	1
1	1	1
1	1	1

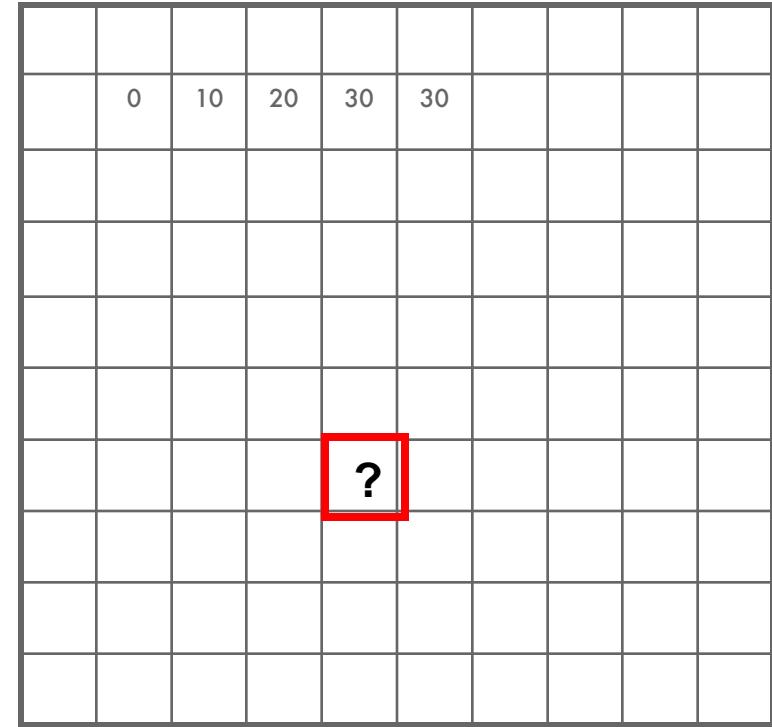
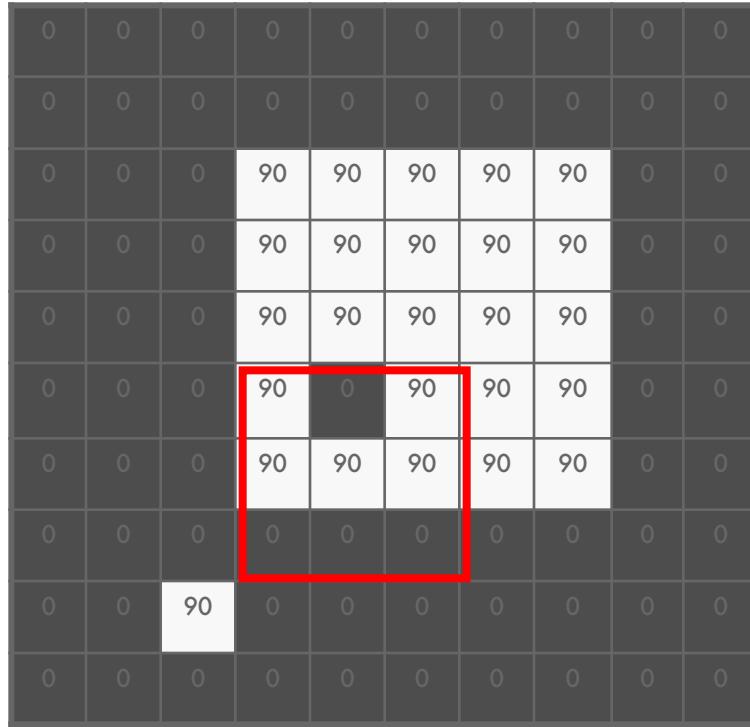


$$g[m, n] = \sum_{k,l} h[k, l] \ f[m + k, n + l]$$

Box Filter

 $f[\cdot, \cdot]$ $g[\cdot, \cdot]$ $h[\cdot, \cdot]$

1	1	1
1	1	1
1	1	1



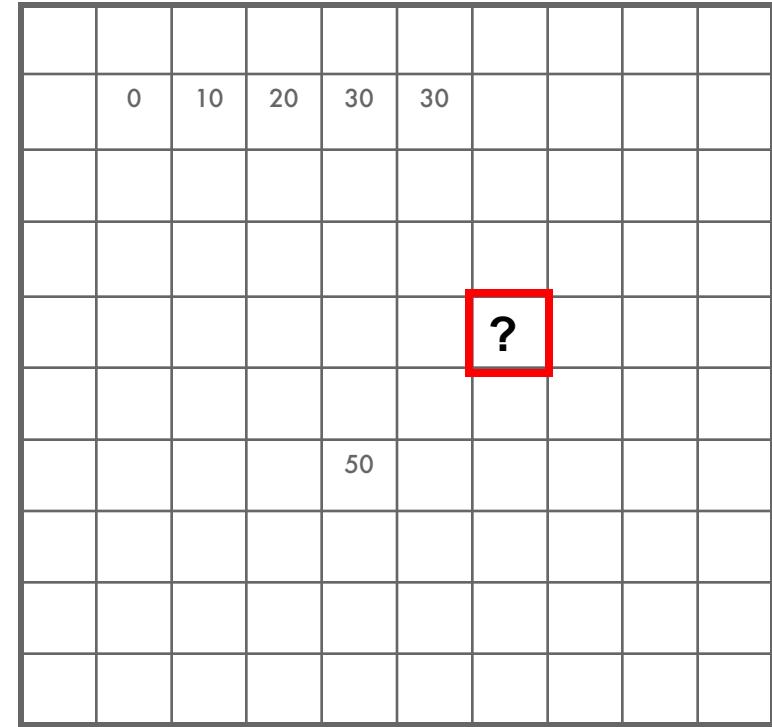
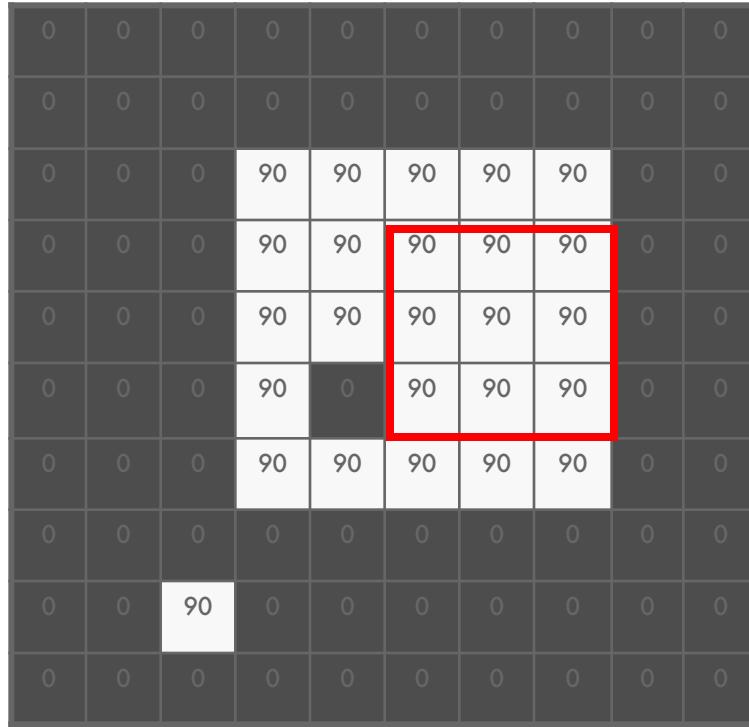
$$g[m, n] = \sum_{k,l} h[k, l] f[m + k, n + l]$$

Credit: S. Seitz
28

Box Filter

 $f[\cdot, \cdot]$ $g[\cdot, \cdot]$ $h[\cdot, \cdot]$

1	1	1
1	1	1
1	1	1



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

Box Filter

 $f[\cdot, \cdot]$ $g[\cdot, \cdot]$ $h[\cdot, \cdot]$

1	1	1
1	1	1
1	1	1

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	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

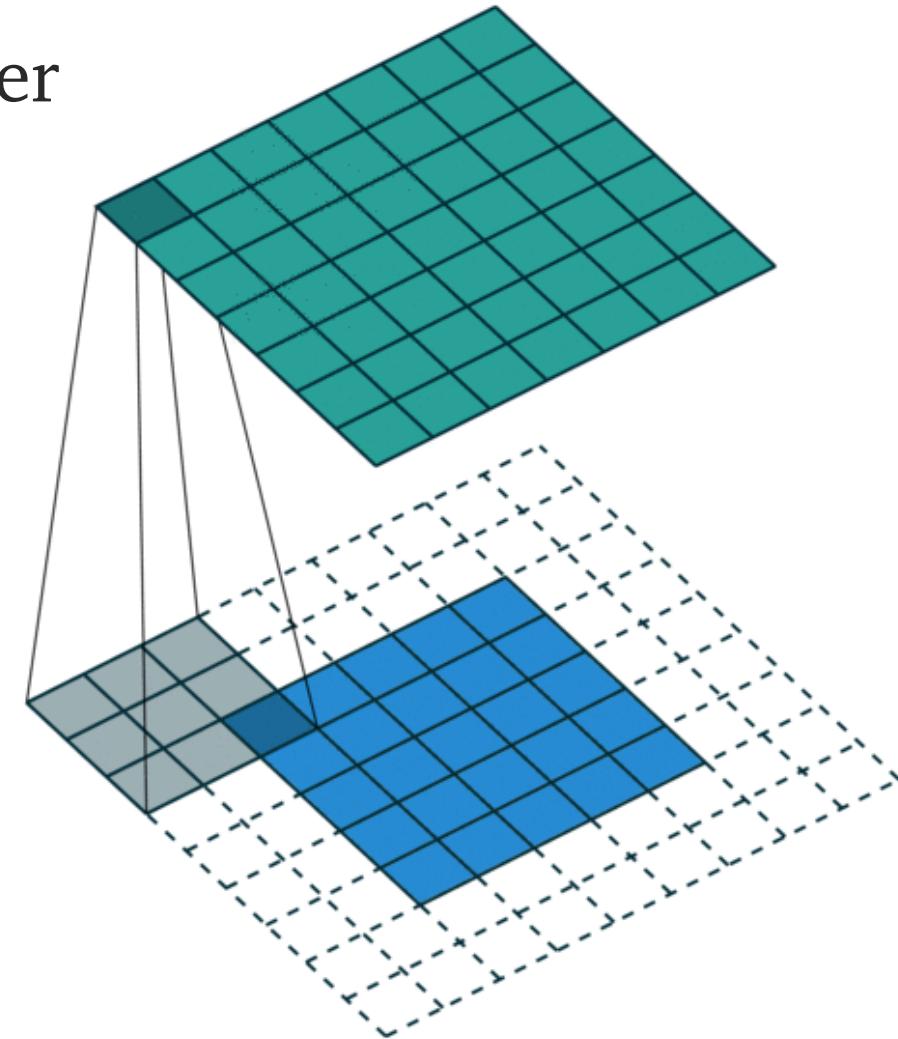
	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

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

Credit: S. Seitz
30

Convolution & Correlation

- Convolution/Correlation is the process of moving a filter mask over an image
- At each point in the image, one computes the sum of products at each location
- Filter is often referred to as the **Kernel or Mask.**
- A function of displacement
- **Convolution:** filter is flipped.
- **Correlation:** filter is not flipped.



Box Filter

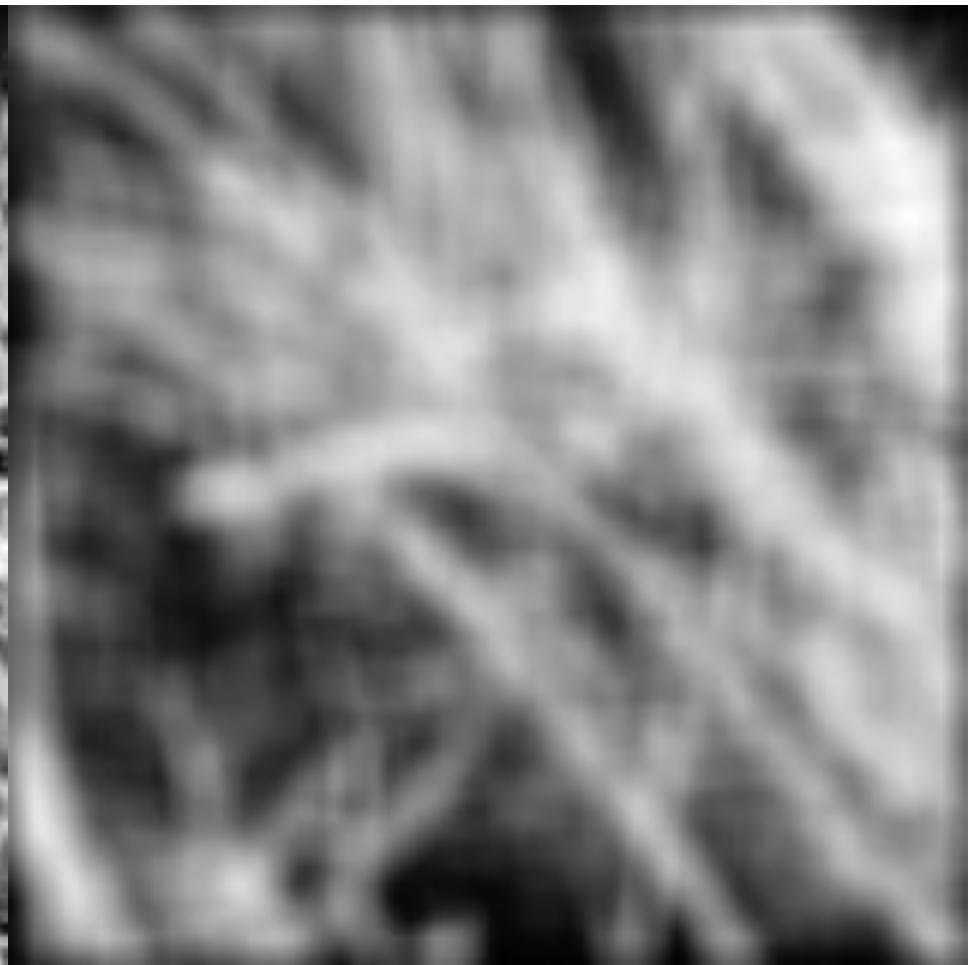


What does it do?

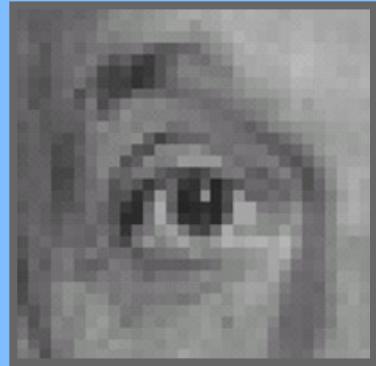
- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)

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

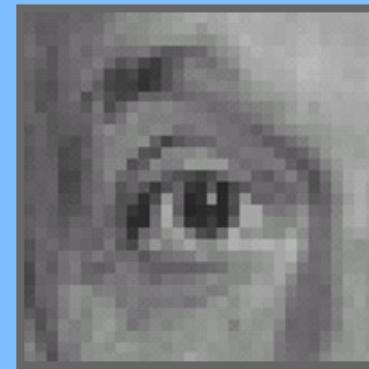
Smoothing with box filter



Predict the filtered outputs



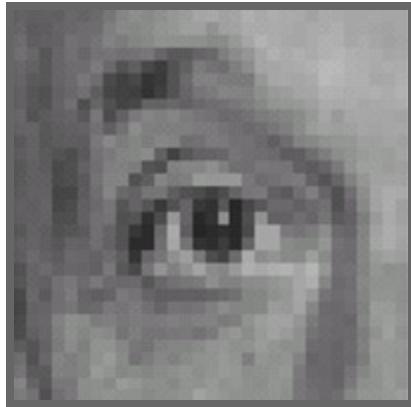
$$\begin{matrix} \text{?} & * & \begin{matrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{matrix} & = ? \end{matrix}$$



$$\begin{matrix} \text{?} & * & \begin{matrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{matrix} & = ? \end{matrix}$$

$$\begin{matrix} \text{?} & * & \begin{matrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{matrix} & - \frac{1}{9} \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix} & = ? \end{matrix}$$

Practice with linear filters

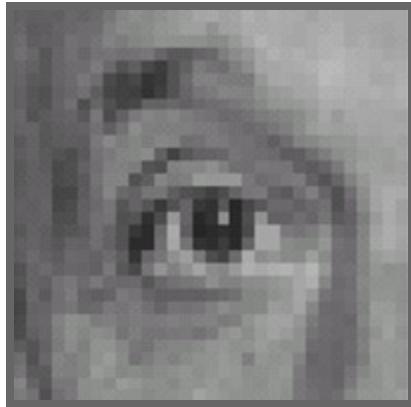


Original

0	0	0
0	1	0
0	0	0

?

Practice with linear filters



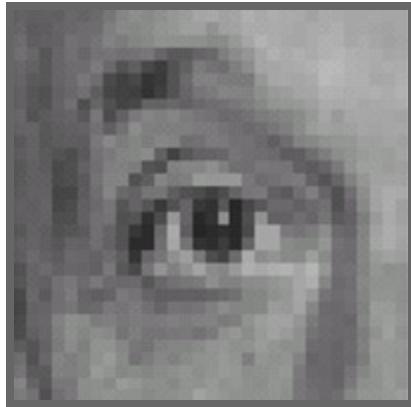
Original

0	0	0
0	1	0
0	0	0



Filtered
(no change)

Practice with linear filters

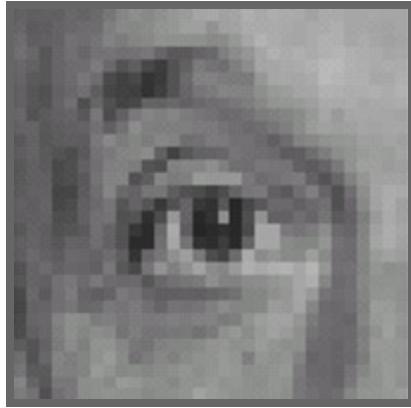


Original

0	0	0
1	0	0
0	0	0

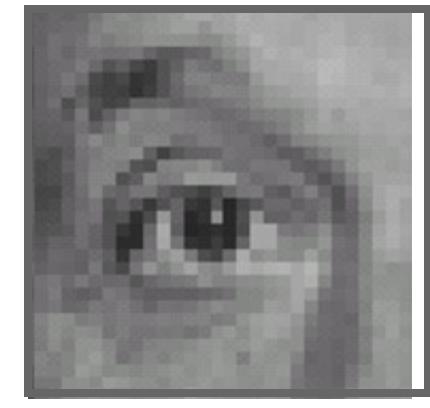
?

Practice with linear filters



Original

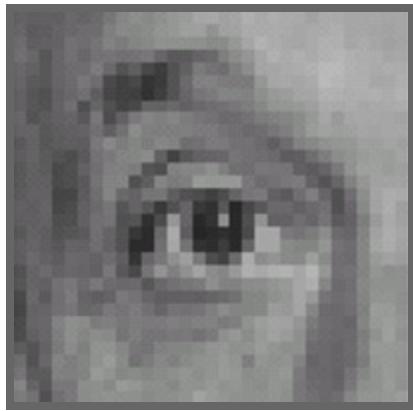
0	0	0
1	0	0
0	0	0



Shifted left
By 1 pixel

Assume using convolution (filter flipped)

Practice with linear filters



Original

0	0	0
0	2	0
0	0	0

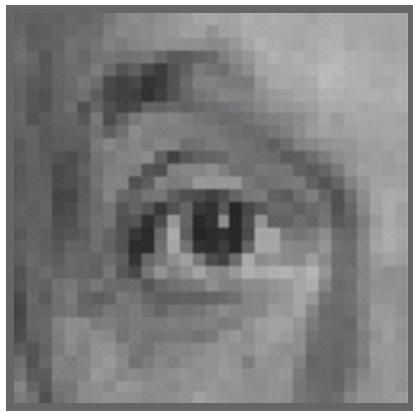
-

$\frac{1}{9}$	1	1	1
1	1	1	1
1	1	1	1

?

(Note that filter sums to 1)

Practice with linear filters

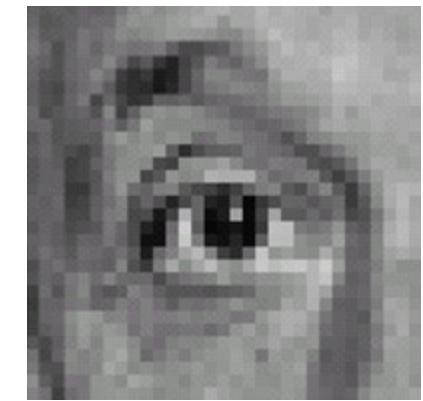


Original

0	0	0
0	2	0
0	0	0

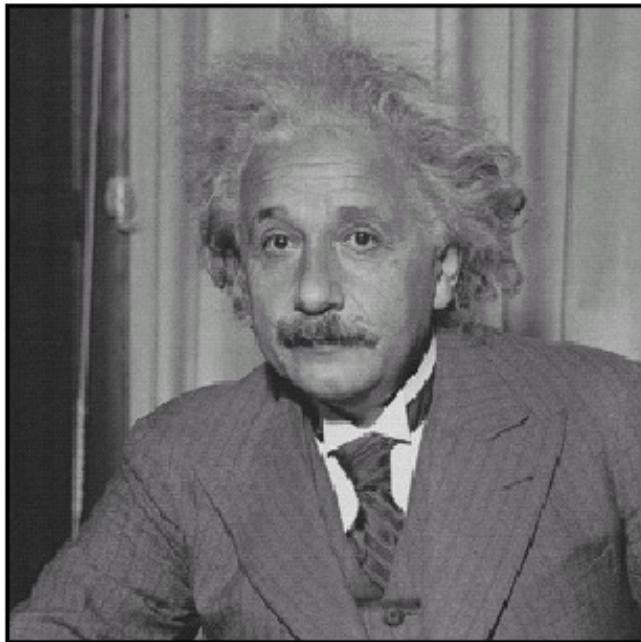
-

$\frac{1}{9}$	1	1	1
1	1	1	1
1	1	1	1

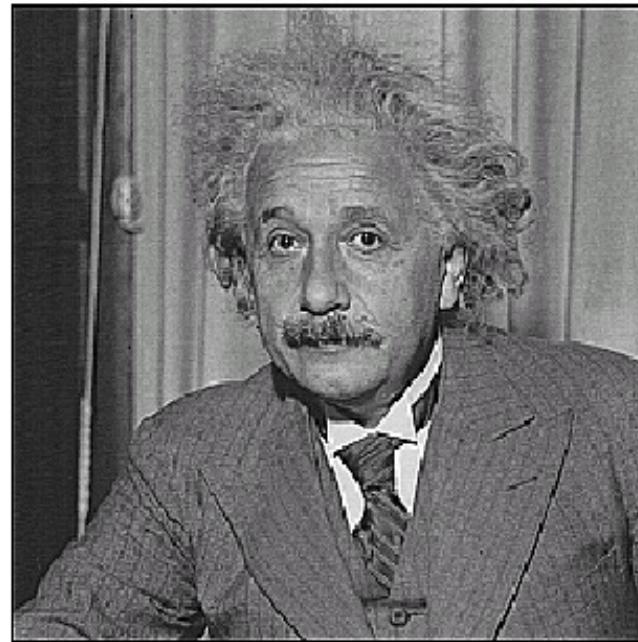


Sharpening filter
Accentuates
differences with local
average

Sharpening



before



after

Correlation (another name for filtering)

$$G[m, n] = \sum_{k,l} H[k, l] F[m + k, n + l]$$

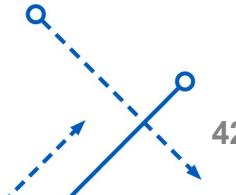
H=Filter F=Signal or Image

• 2D Convolution

- PyTorch implements correlation as convolution
- import torch.nn.functional as F
- `G = F.conv2d(I, f)`

$I = \text{image}$ $f = \text{filter}$

$$G[m, n] = \sum_{k,l} H[k, l] F[m - k, n - l]$$



Correlation vs. Convolution

- 2D Correlation

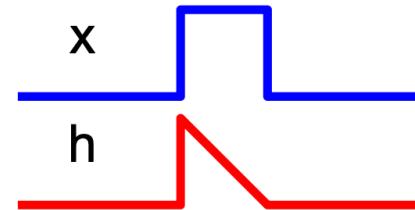
$$h[m, n] = \sum_{k, l} f[k, l] I[m + k, n + l]$$

- 2D Convolution

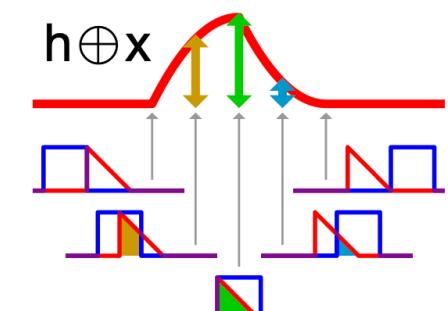
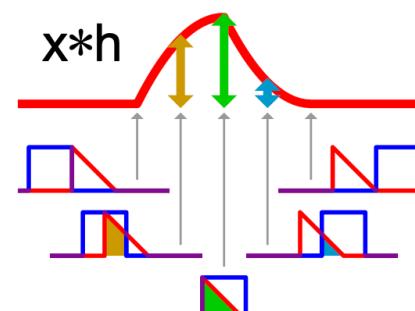
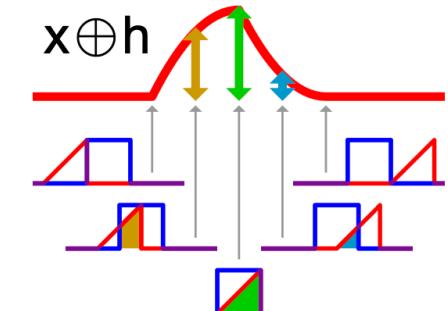
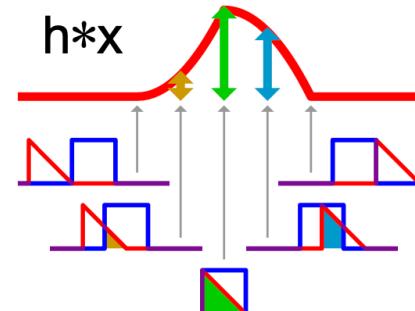
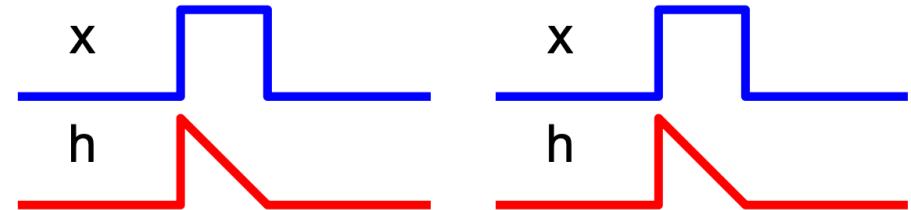
$$h[m, n] = \sum_{k, l} f[k, l] I[m - k, n - l]$$

- 1D Case

Cross-Correlation



Convolution



Effect on the filter

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

When $k = l = -1 \rightarrow (m-1, n-1)$

1	2	3
4	5	6
7	8	9

EXERCISE

$$h[m, n] = \sum_{k, l} f[k, l] I[m - k, n - l]$$

When $k = l = -1 \rightarrow (m+1, n+1)$

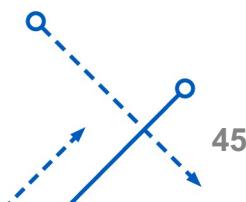
1	2	3
4	5	6
7	8	9

Convolution Filters
are rotated by 180
degrees

What Properties do they have?

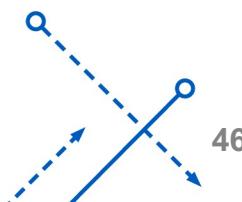


- Both extract information from the image
- Both are shift or translation Invariant
- Both are linear (a linear combination of neighbors)
- Only the **Convolution** is Associative
 - $F * (G * I) = (F * G) * I$



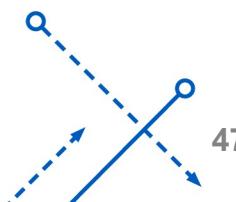
When to use which?

- Correlation
 - Applying a template or filter to an image
 - Measuring Similarity
 - When we don't care of it is associative
- Convolution
 - Applying an operation to an image (filtering)
 - When we want association



What to keep in mind

- If we say we are applying a **correlation** and we provide a filter, assume it is a correlation filter.
- If we say we are applying a **convolution** and we provide a filter, assume it is a convolution filter. No rotation is needed.
- You can get the same point, with either approach



Correlation filtering

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

Say the averaging window size is $(2k+1) \times (2k+1)$:

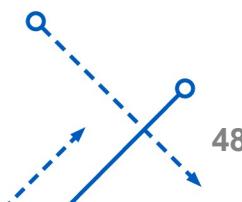
$$G[i, j] = \frac{1}{(2k+1)^2} \underbrace{\sum_{u=-k}^k \sum_{v=-k}^k}_{\text{Attribute uniform weight to each pixel}} F[i + u, j + v]$$

Attribute uniform weight to each pixel *Loop over all pixels in neighborhood around image pixel $F[i,j]$*

Now generalize to allow different weights depending on neighboring pixel's relative position:

$$G[i, j] = \sum_{u=-k}^k \sum_{v=-k}^k H[u, v] \underbrace{F[i + u, j + v]}_{\text{Non-uniform weights}}$$

Non-uniform weights



Correlation filtering

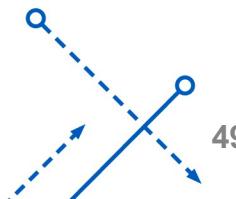
$$G[i, j] = \sum_{u=-k}^k \sum_{v=-k}^k H[u, v] F[i + u, j + v]$$

More formally, it is called cross-correlation, i.e.,

$$G = H \otimes F$$

Filtering an image: replace each pixel with a linear combination of its neighbors.

The filter is also called “kernel”, “mask”, or a “window”.



More Formal/Strict Terminology

- Cross-correlation:
 - Often referred as **correlation** in computer vision.

$$G[m, n] = \sum_{k,l} H[k, l] \ F[m + k, n + l]$$

- Correlation (in statistics):

$$\text{corr}(X, Y) = \frac{\text{cov}(X, Y)}{\sigma_X \sigma_Y} = \frac{\text{E}[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y}$$

- Reading more:
 - Chen Wang, et al. "Kernel Cross-correlator." In Proceedings of the AAAI Conference on Artificial Intelligence, 2018.
 - Convolution vs Cross-correlation
 - Section 2.2 in “Chen Wang, Kernel learning for visual perception”. PhD thesis.

Properties

- Commutative:

$$x \otimes y = y \otimes x$$

- Conceptually no difference between filter and signal
- But particular filtering implementations might break this equality

- Associative:

$$x \otimes (y \otimes z) = (x \otimes y) \otimes z$$

- Often apply several filters one after another: $x \otimes y_1 \otimes y_2 \otimes y_3$
- This is equivalent to applying one filter: $x \otimes (y_1 \otimes y_2 \otimes y_3)$

- Distributes over addition:

$$x \otimes (y + z) = (x \otimes y) + (x \otimes z)$$

- Scalars factor out:

$$ax \otimes y = x \otimes ay = a(x \otimes y)$$

- Identity (unit impulse), e.g., $e = [0, 0, 1, 0, 0]$,

$$x \otimes e = x$$

Key properties of Linear Filters

- Assume $f(x)$ is image filtering: $f(x) = x \otimes w$.

- Linearity (superposition property):

$$f(a \cdot x + b \cdot y) = af(x) + bf(y)$$

- Linear filter is **equivariant** (not **invariant**) to translation.

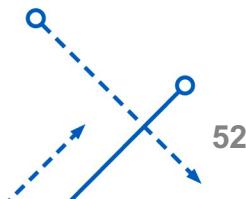
- Assume $T(x)$ is image shift (translation):

- **Equivariance:**

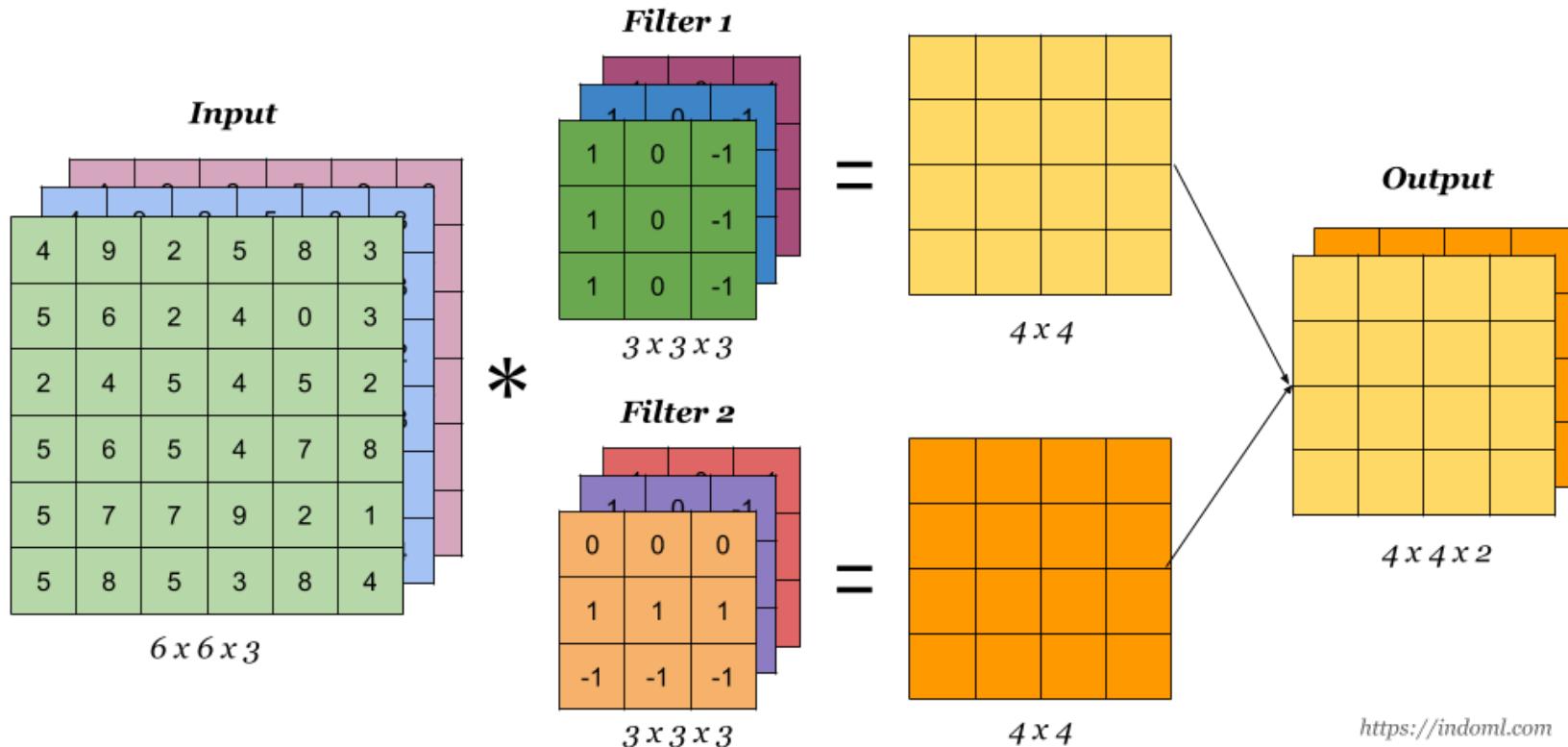
$$f(T(x)) = T(f(x))$$

- **Invariance:**

$$f(T(x)) = f(x)$$



Multi-channel correlation



Examples:

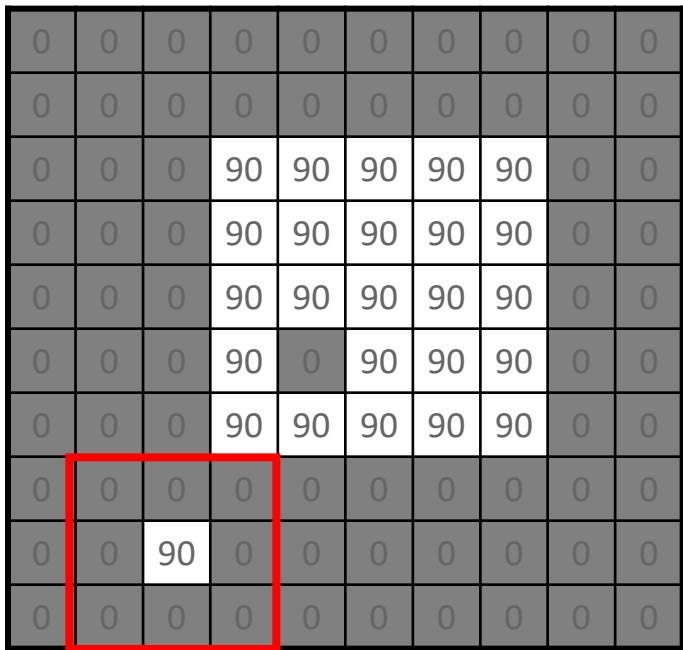
```
>>> # With square kernels and equal stride
>>> filters = torch.randn(8, 4, 3, 3)
>>> inputs = torch.randn(1, 4, 5, 5)
>>> F.conv2d(inputs, filters, padding=1)
```

What is shape of the output
in the left example?

(1, 8, 5, 5)

Gaussian filter

- What if we want nearest neighboring pixels to have the most influence on the output?



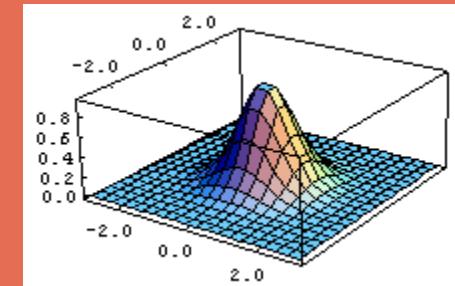
$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

$H[u, v]$

This kernel is an approximation of a Gaussian function:

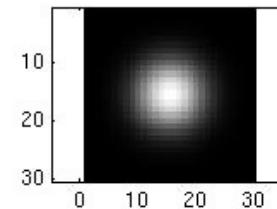
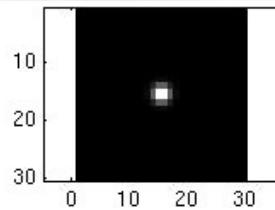
$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



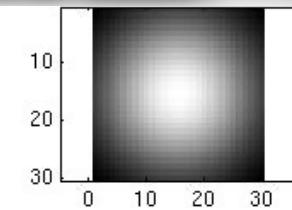
$F[x, y]$

Smoothing with a Gaussian

- Parameter σ is the “scale” / “width” / “spread” of the Gaussian kernel, and controls the amount of smoothing.

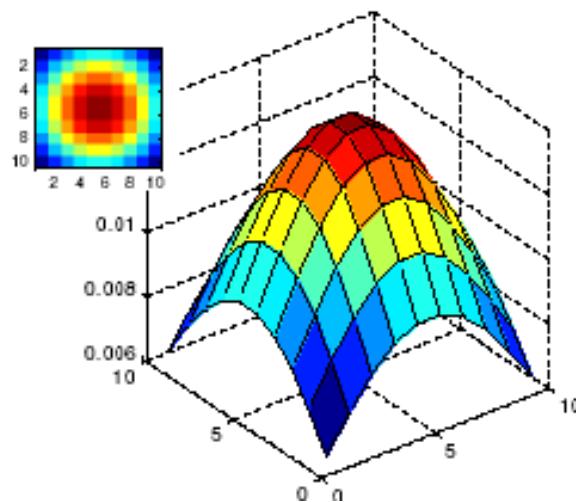


...

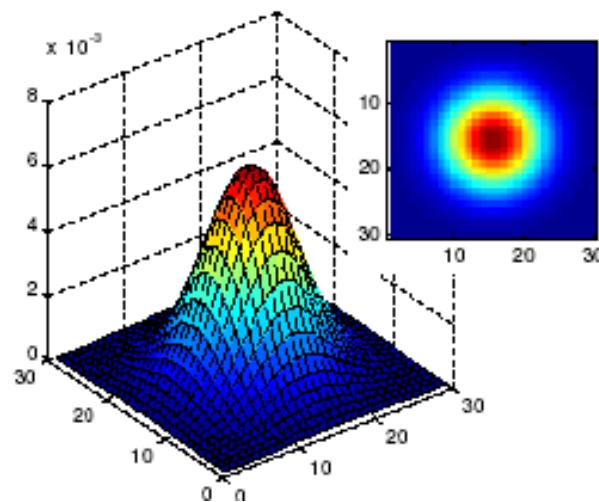


Gaussian filters

- What parameters matter here?
- **Size** of kernel / mask
 - Gaussian function has infinite support, but discrete filters use finite kernels



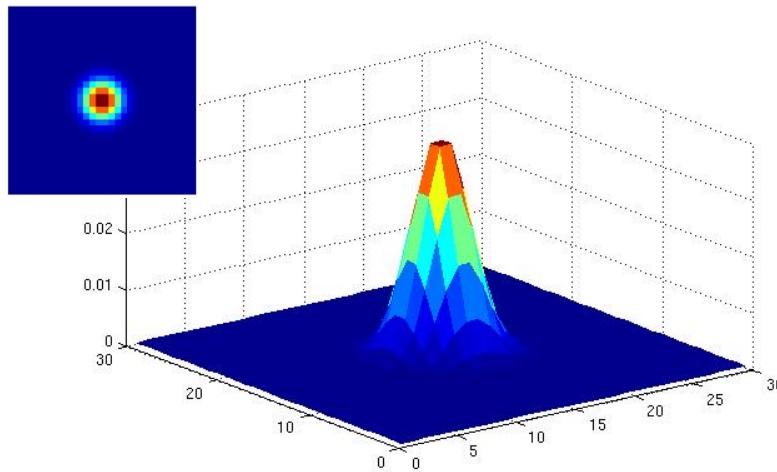
$\sigma = 5$ with 10×10 kernel



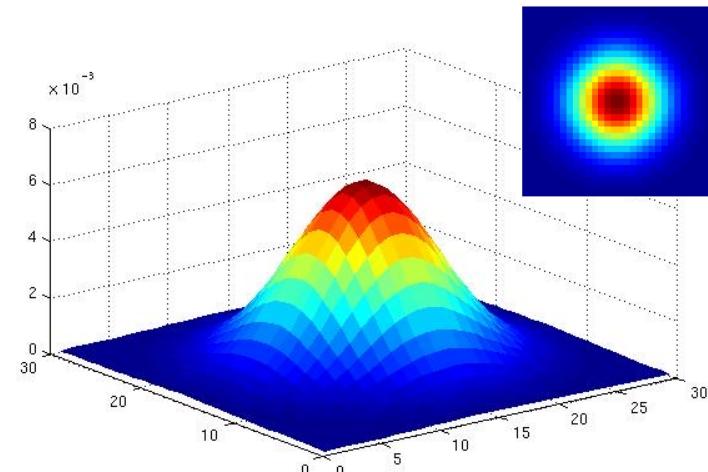
$\sigma = 5$ with 30×30 kernel

Gaussian filters

- **Variance:** determines extent of smoothing



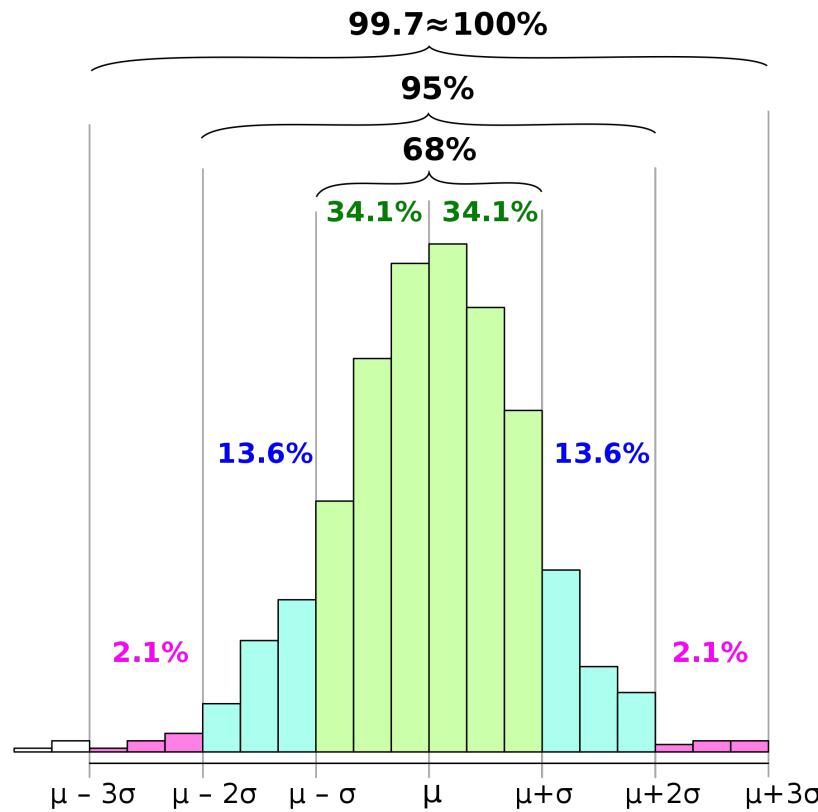
$\sigma = 2$ with
30 x 30
kernel



$\sigma = 5$ with
30 x 30
kernel

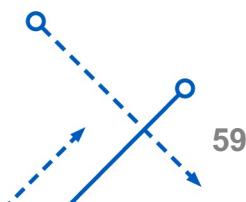
How big should the filter be?

- Values at edges should be near zero
- Rule of thumb for Gaussian filter:
 - set filter half-width to about 3σ

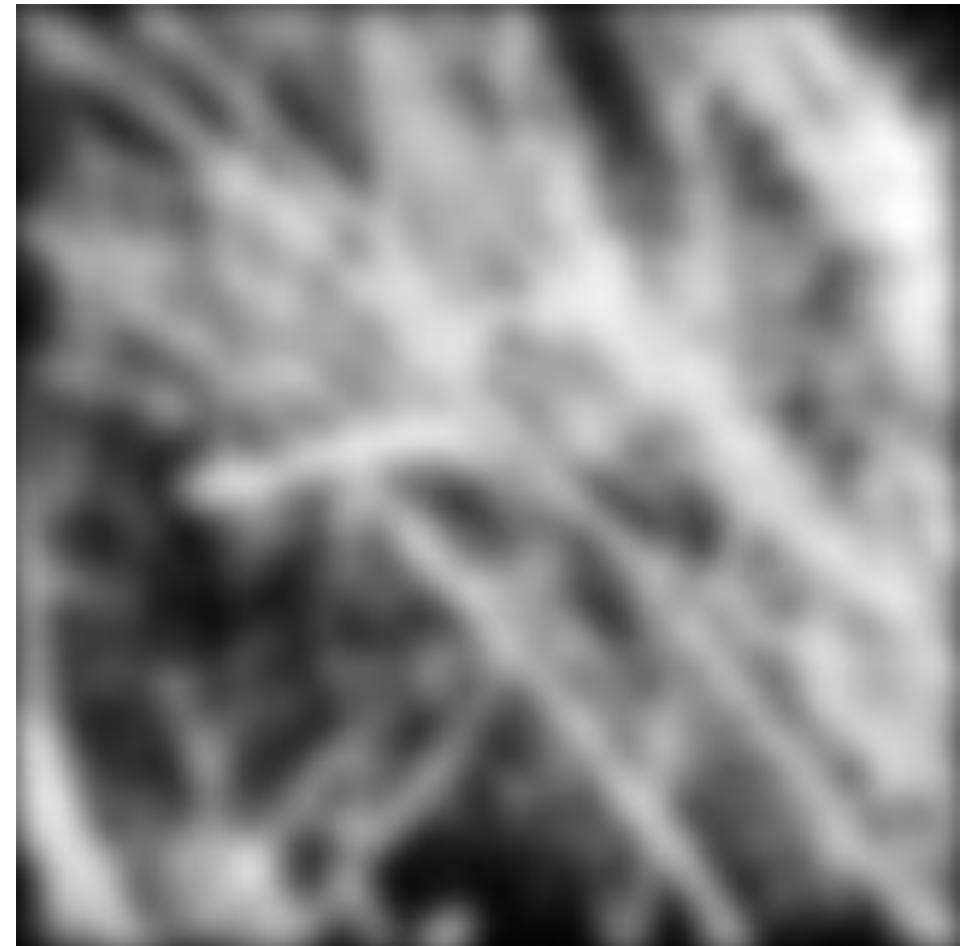


Gaussian filters

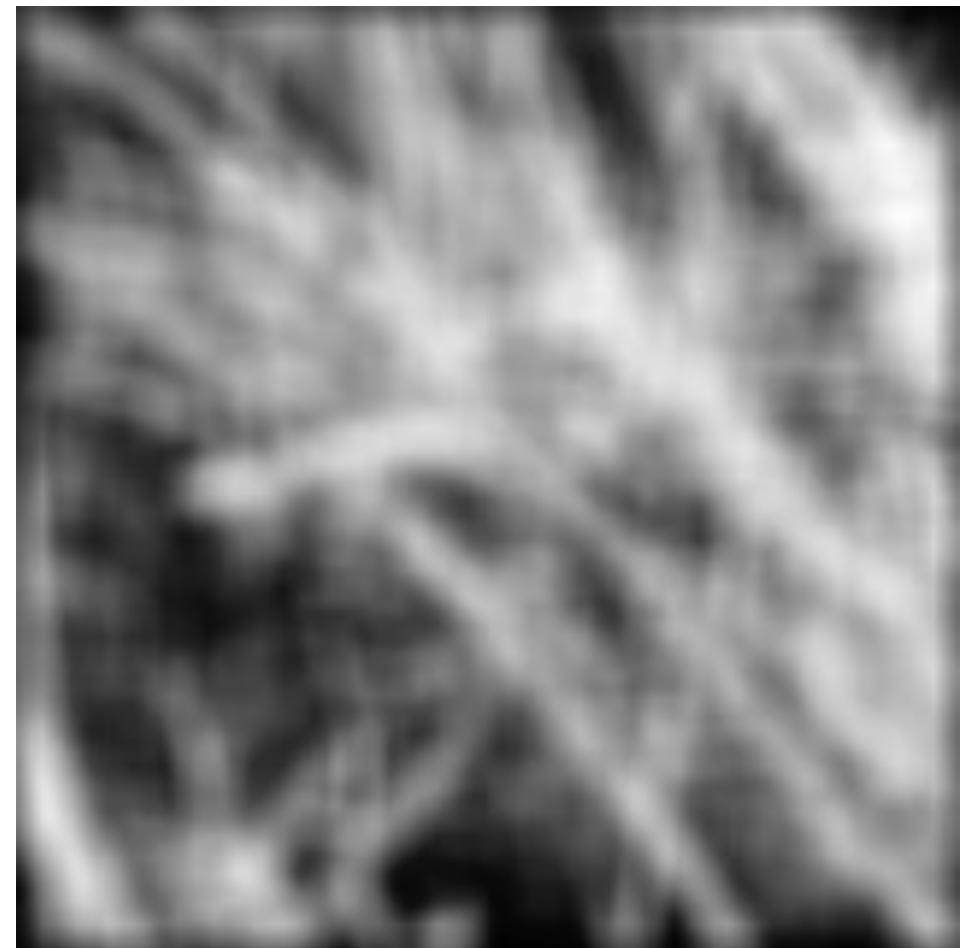
- Remove “high-frequency” components from image
 - Low-pass filter: Images become smoother.
- Recap of frequency in a signal (image)
 - High-frequency components
 - Fine details and edges
 - Low-frequency components
 - Large-scale structures and smooth regions



Smoothing with Gaussian filter



Smoothing with box filter



Gaussian filters

- 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

$$\begin{aligned} G_\sigma(x, y) &= \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2 + y^2}{2\sigma^2}} \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{x^2}{2\sigma^2}} \right) \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{y^2}{2\sigma^2}} \right) \end{aligned}$$

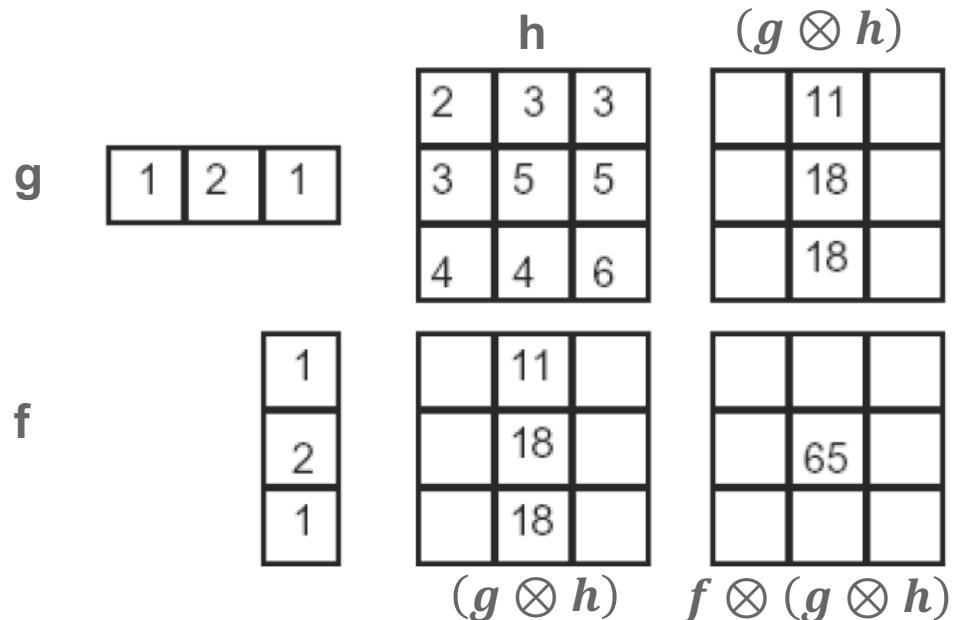
The 2D Gaussian can be expressed as the product of two functions, one a function of x and the other a function of y

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

Separability

- In some cases, filter is separable, (factor into two steps):

$$f \otimes (g \otimes h) = (f \otimes g) \otimes h$$



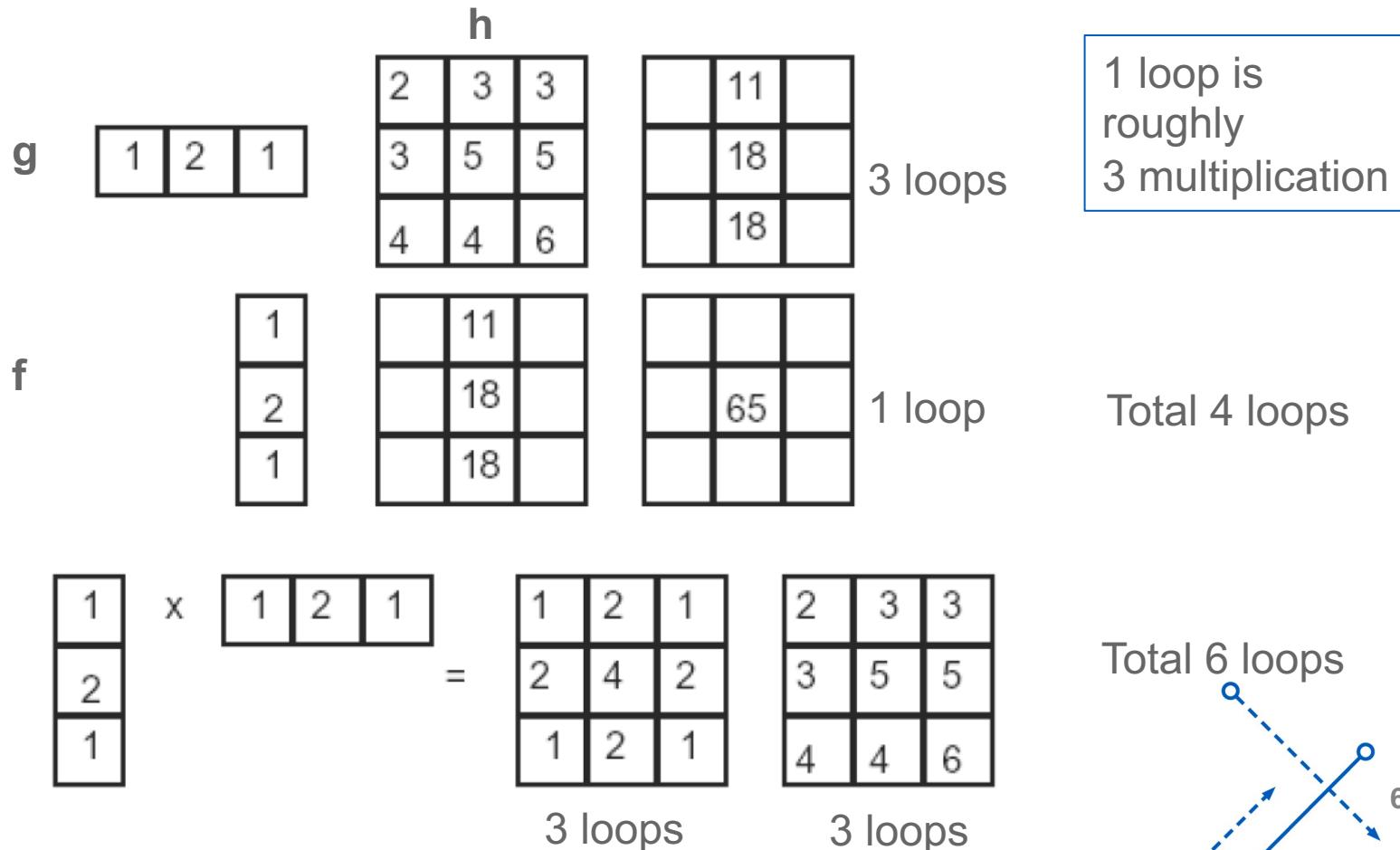
$$\begin{matrix} 1 \\ 2 \\ 1 \end{matrix} \times \begin{matrix} 1 & 2 & 1 \end{matrix} = \begin{matrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{matrix} \quad \begin{matrix} 2 & 3 & 3 \\ 3 & 5 & 5 \\ 4 & 4 & 6 \end{matrix}$$

$(f \otimes g)$ h

What is the computational complexity advantage for a separable filter of size $k \times k$, in terms of **number of operations per output pixel**?

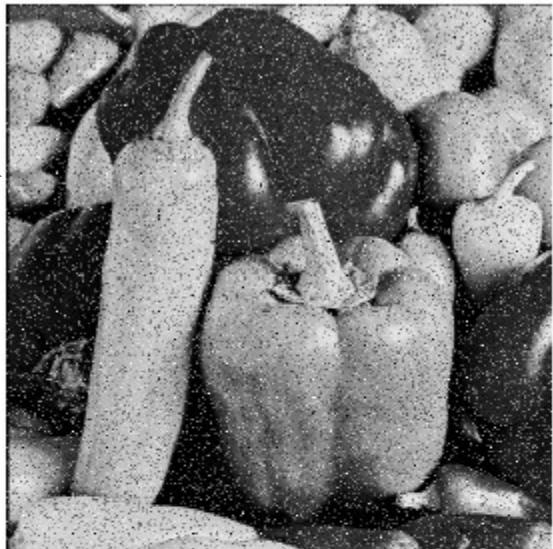
Separability

- **Advantage:** much reduced computational cost.
- **Disadvantage:** requires an extra ram **memory** to store the intermediate image, problematic in certain applications.

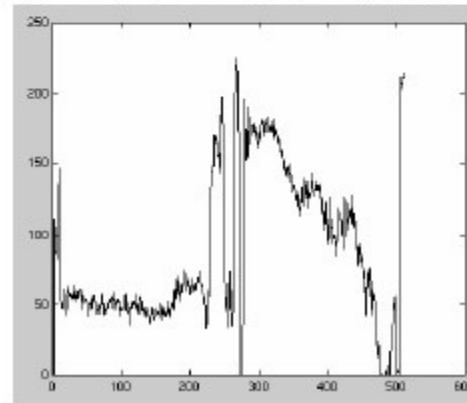
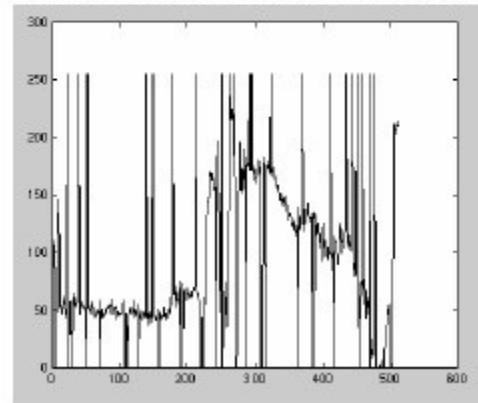


Median filter

Salt and pepper noise



Median filtered

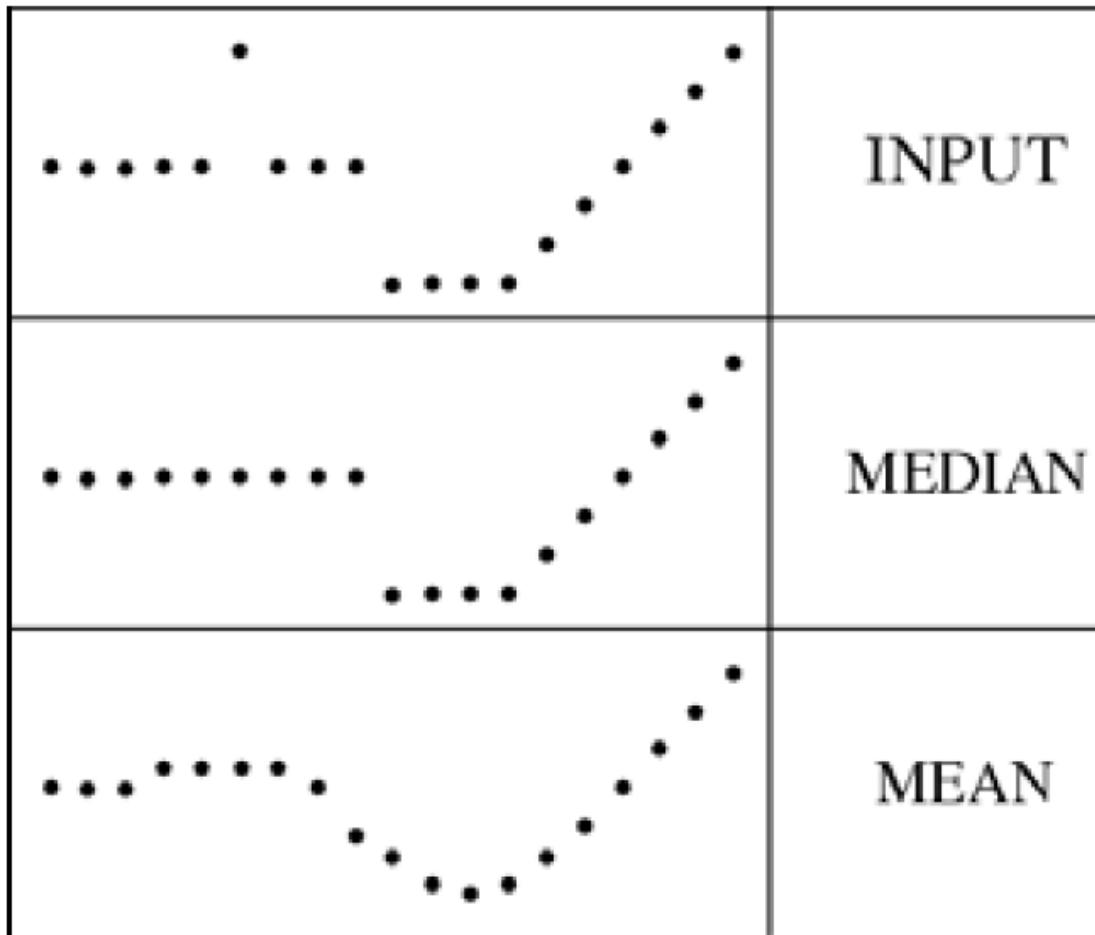


Plots of a row of the image

Source: M. Hebert

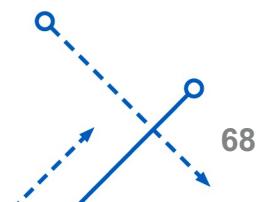
Median filter

- Median filter is edge preserving
- It doesn't introduce new intensities, which is often expected.



Content

- Filtering
 - Linear filters
 - Correlation and Convolution
 - Equivariance, Invariance
 - Smoothing, Gaussian Filter, Median filter



68