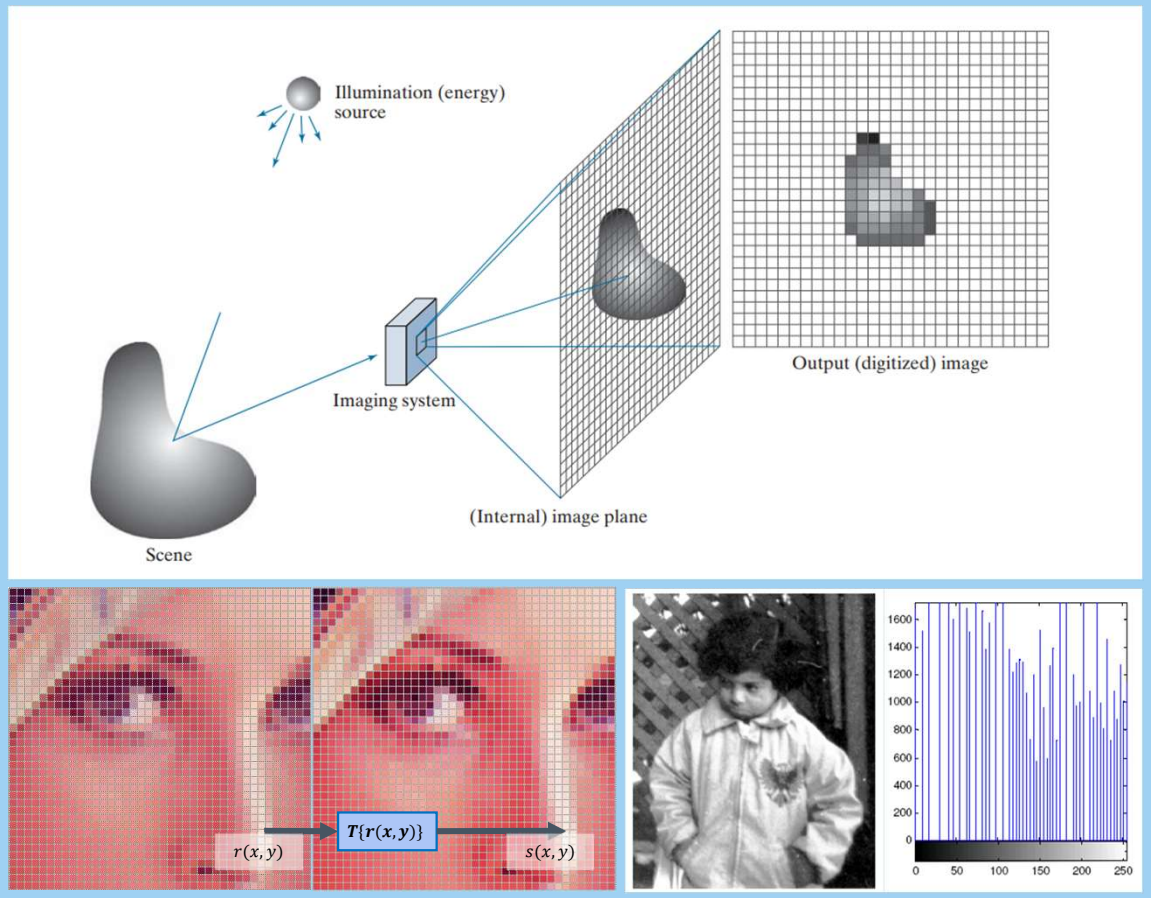
Lecture 3
Filtering

PREVIOUSLY

- Image Formation
  - Pixels
  - Sampling and Quantization
- Pixel (point)-based Processing
- Image Histograms
  - Histogram Equalization
- Recommended Reading
  - [Gonzalez&Woods] Chapter 2 & 3

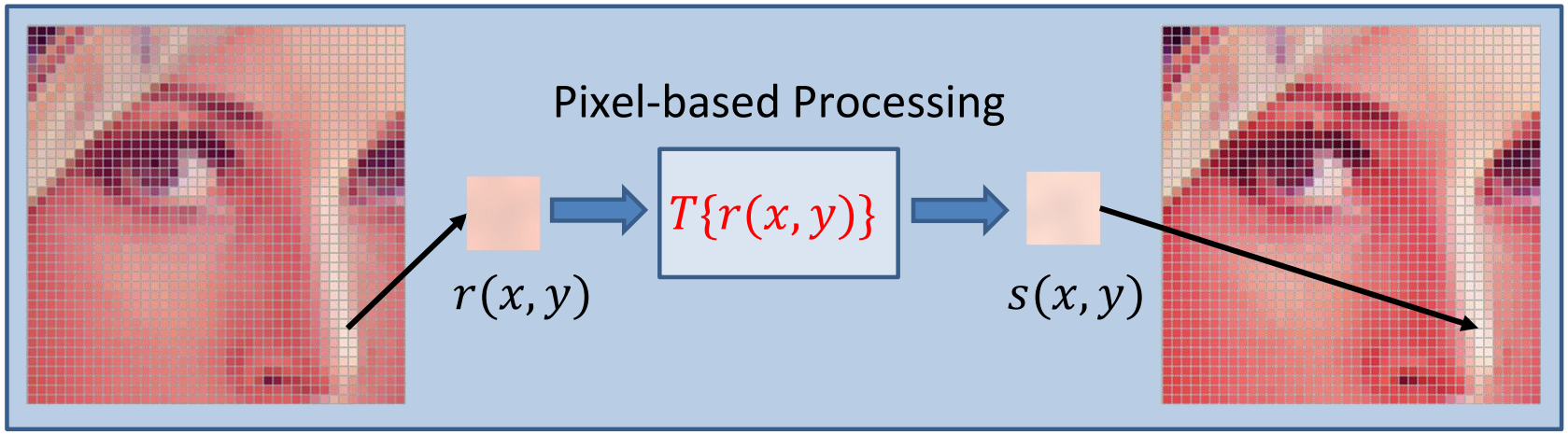


Lecture Outline

- Image Filtering
- Correlation and Convolution
- Image Sharpening
- Non-linear filters

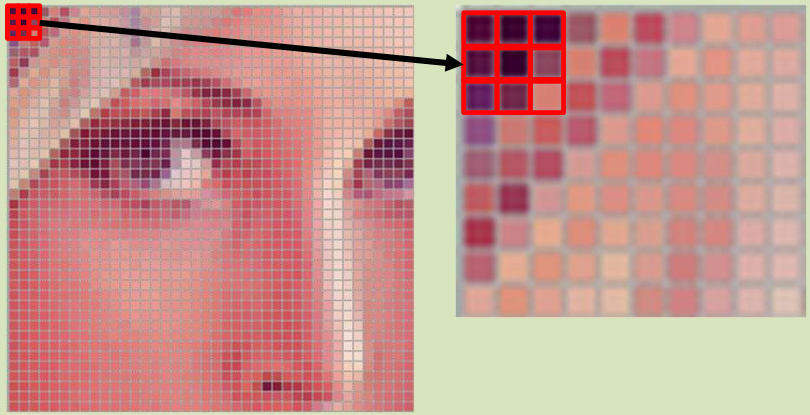
Image Filtering

Previously



This lecture:

Neighbourhood-based Processing



Applications of Filtering

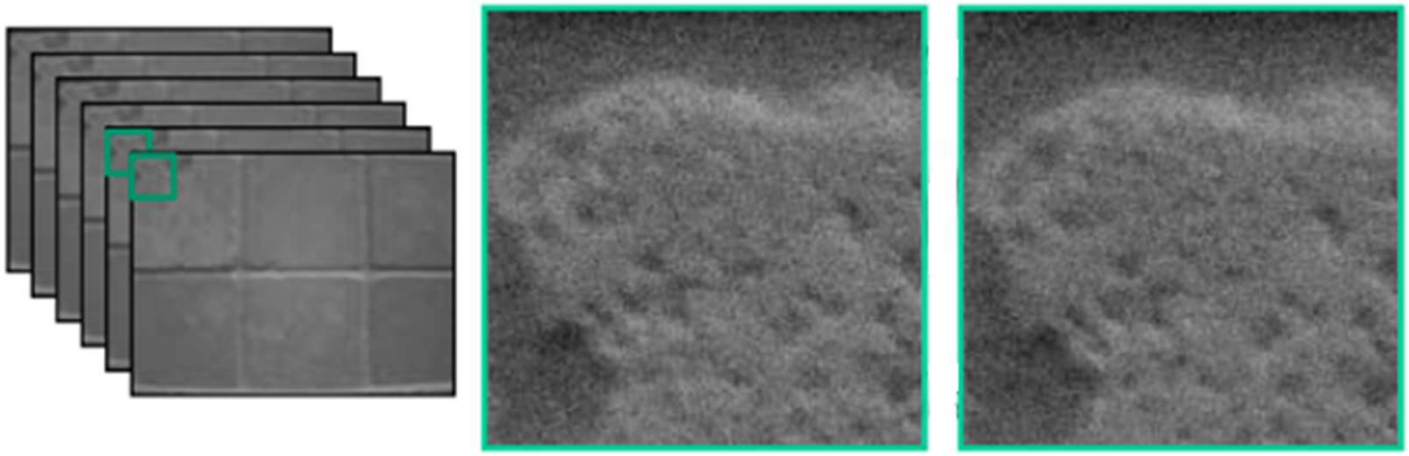
De-noising

Salt and pepper noise

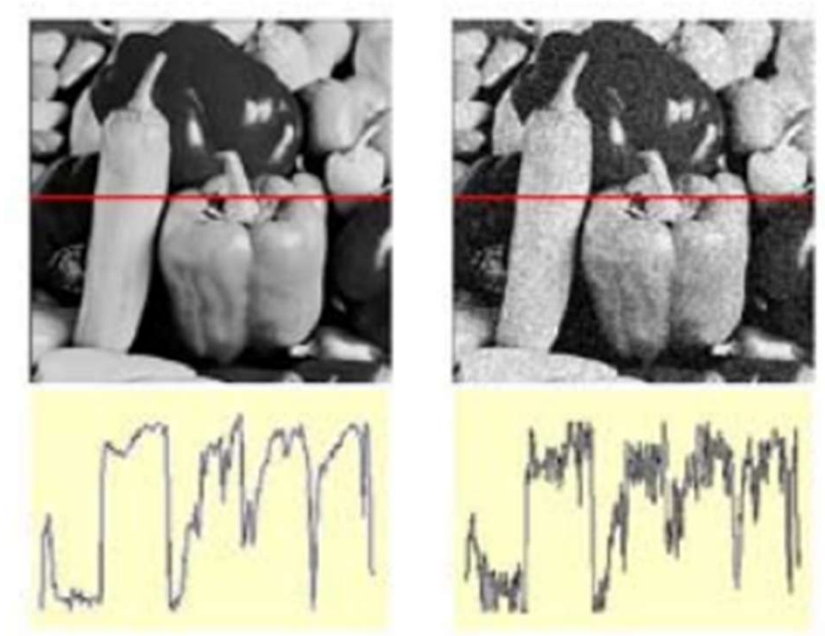
Super-resolution

Motivation: Noise Reduction

› Capturing multiple images of the same static scene will not result in identical images
  – Likely: Environmental changes, sensor noise, camera shake, etc.



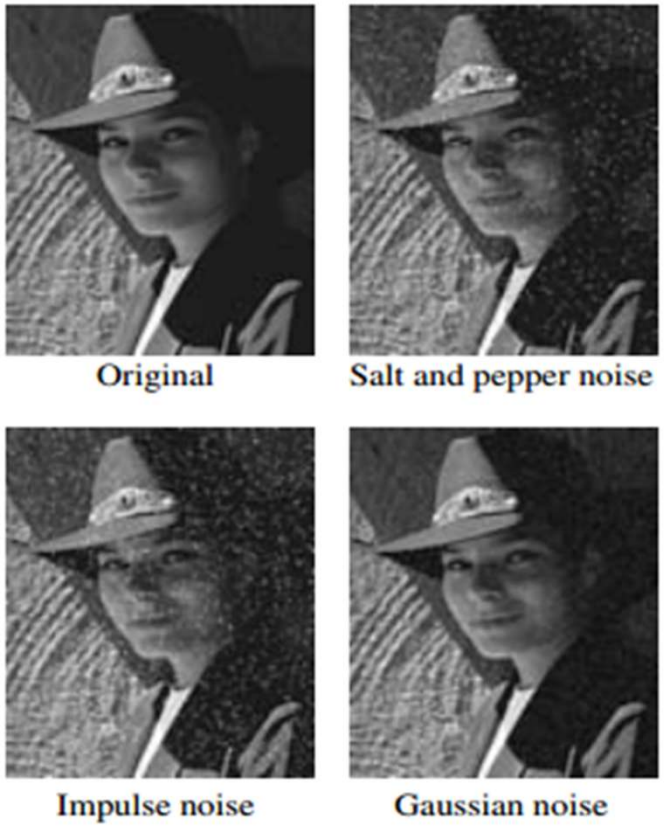
Motivation: Noise Reduction



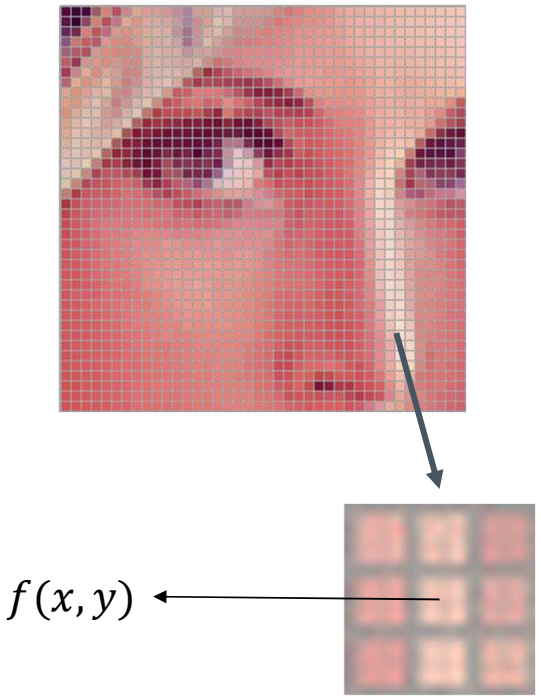
› Sometimes noises can be introduced during transmission of signals or compression of data

Common Types of Noise

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



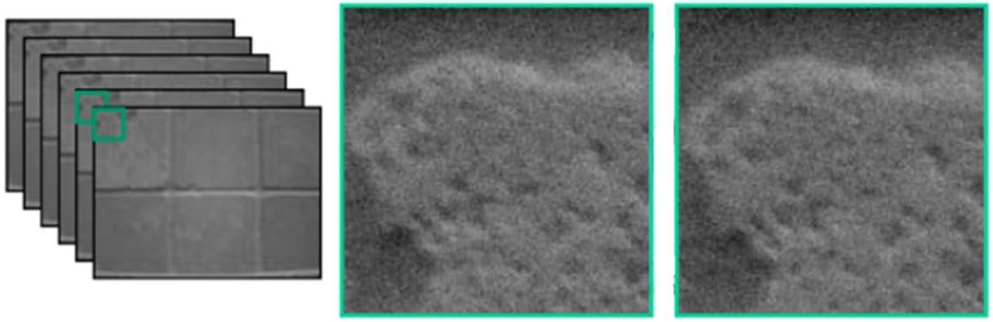
Neighborhood Processing a.k.a. Filtering



- Idea: Modify the value of the pixel $f(x, y)$ based on a small neighbourhood of pixels surrounding it
- If we wish to "soften" the noise in the image, how should we modify?

A. Get minimum pixel
B. Get maximum pixel
C. Get average pixel
D. Use a preset value

Solution: Noise Reduction



› Idea: Replace each pixel with an average of all the values in its neighbourhood

› Assumptions:

- Expect pixels to be “like” their neighbours
- Expect noise processes to be independent from pixel to pixel

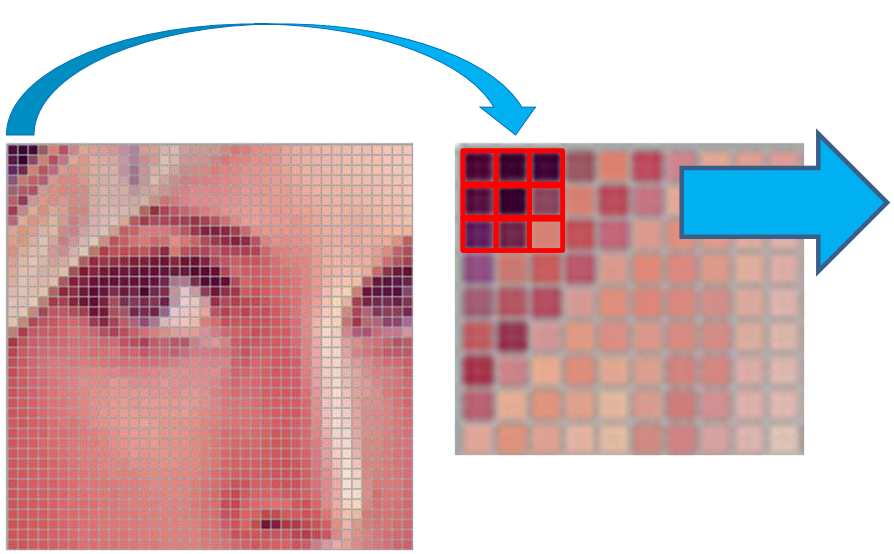
Result

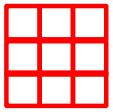

How can we represent this idea of averaging pixels in the neighbourhood area of each pixel?

“Masking” or Filtering

› Run a “mask” or “filter” across the entire image

› Mask corresponds to the neighbourhood that we wish to process



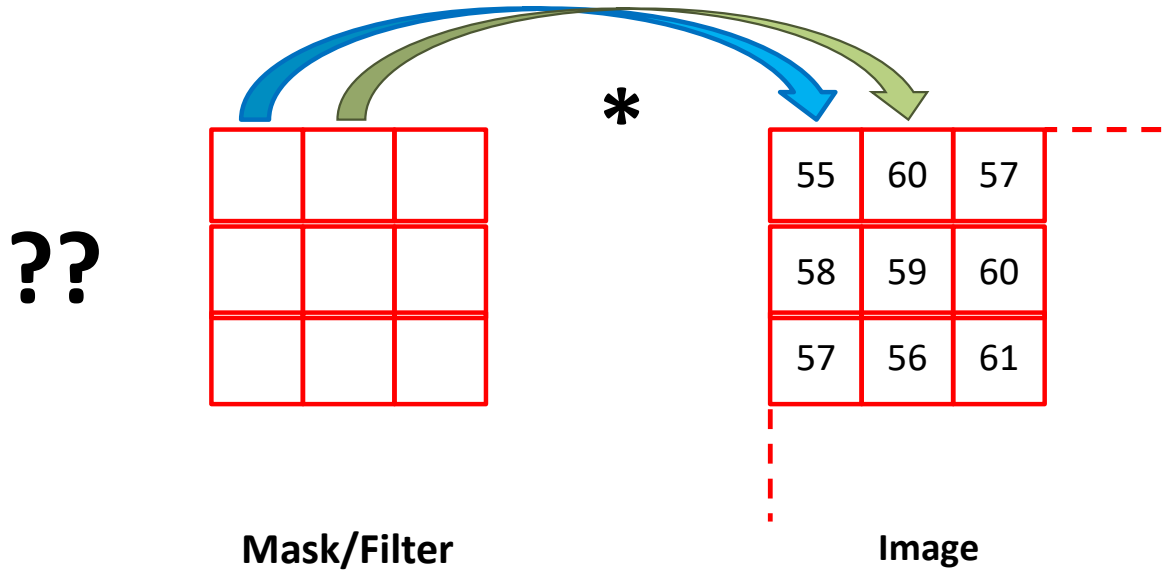


3-by-3 size mask

1. Find the average of all 9 pixels
2. Store the output
3. Slide the mask horizontally along the row.
4. Repeat

“Masking” or Filtering

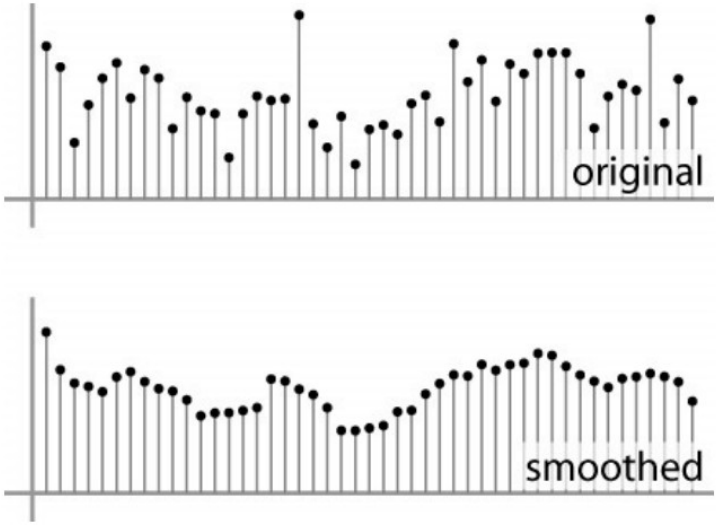
› Let’s say we multiply element-by-element the 3x3 mask/filter against a 3x3 part of the image, what are the values in the mask to achieve the **averaging** effect?



Process in 1-D

› Let’s replace each pixel with an average of all the values in its neighbourhood (also called “moving average”)

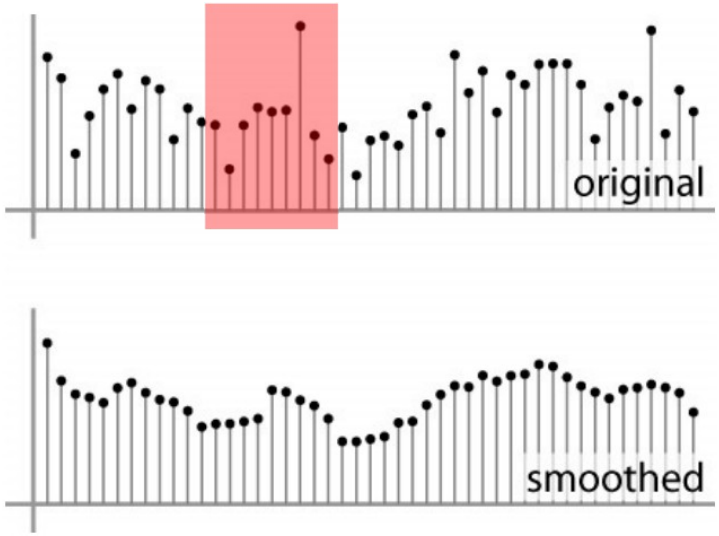
› 1D example:



Moving Average

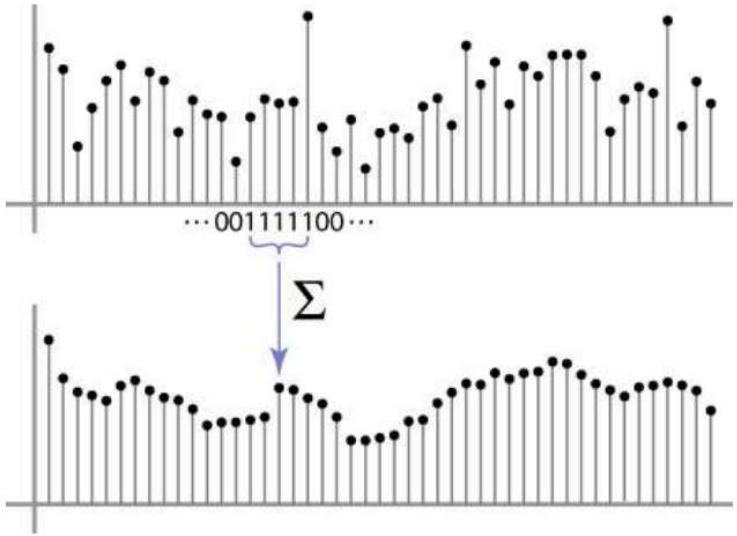
› Let’s replace each pixel with an average of all the values in its **neighbourhood**

› What’s this neighbourhood?



Moving Average

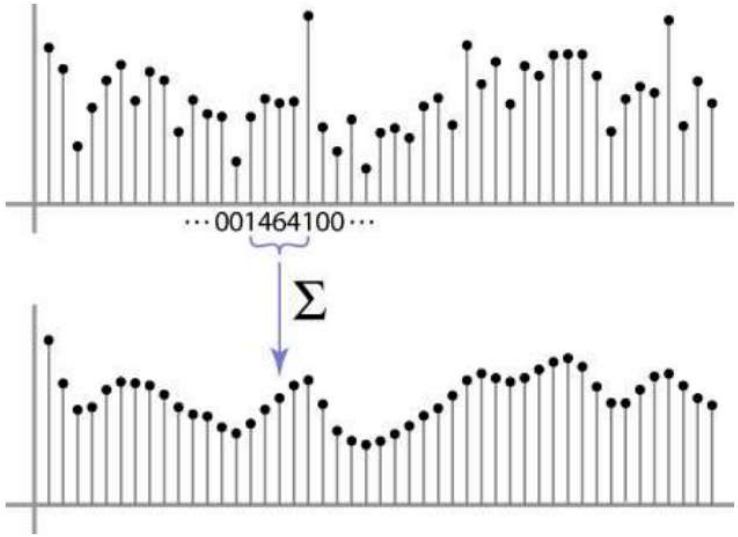
- Moving average has equal uniform weights in the neighbourhood
- Example: [1 1 1 1 1] / 5



Weighted Moving Average

› Moving average with non-uniform weights

› Example: [1 4 6 4 1] / 16



Moving Average in 2-D

$F[x, y]$

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 in 2-D

$F[x, y]$

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 in 2-D

$F[x, y]$

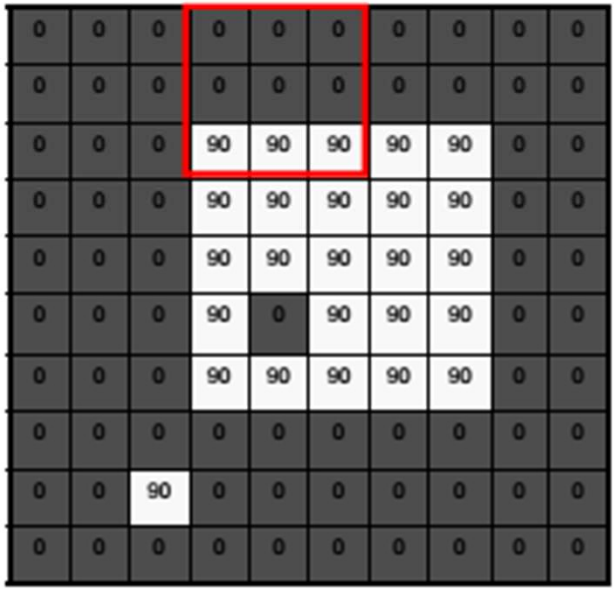
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

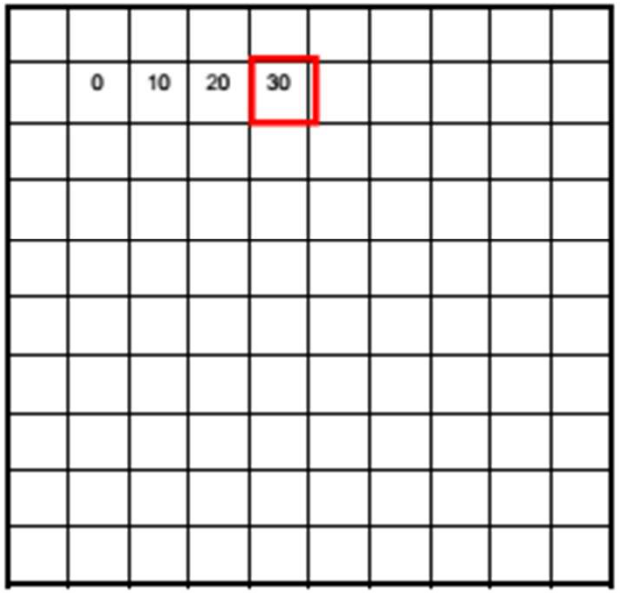
	0	10	20						

Moving Average in 2-D

$F[x, y]$



$G[x, y]$



Finish: Moving Average in 2-D

$F[x, y]$

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	

How does the filtered image look visually?

Correlation and Convolution

Correlation Filtering

› Say the averaging window size is $2k$+1×$2k$+1:

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

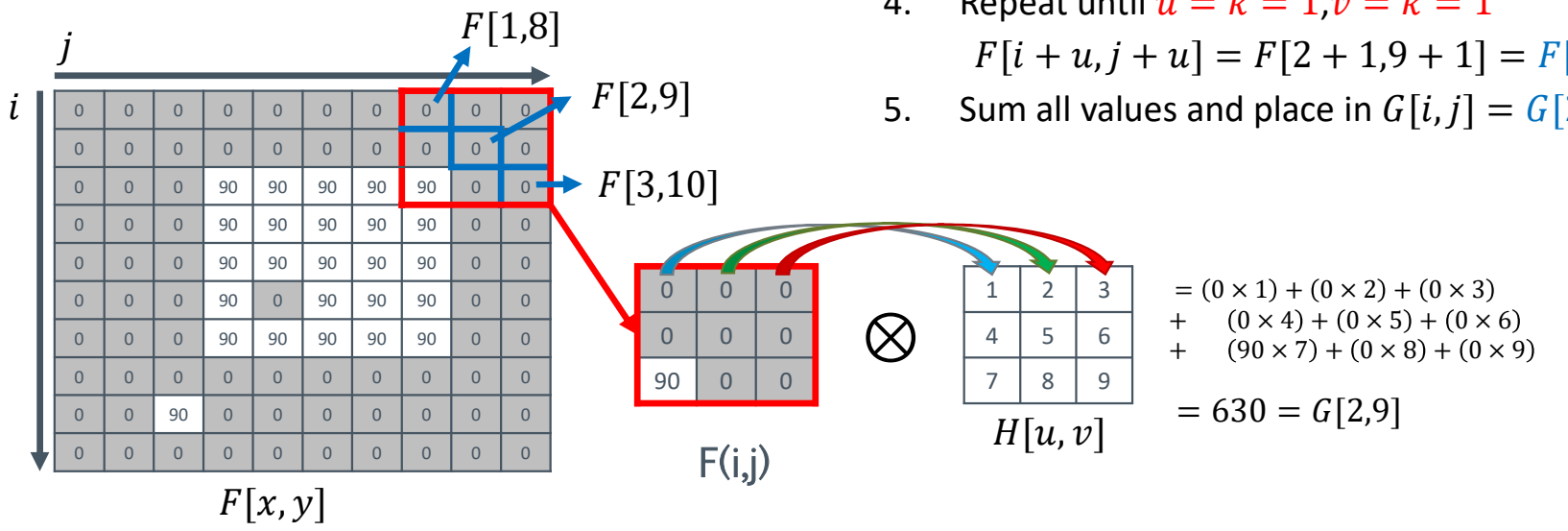
› Generalize to allow different weights depending on neighbouring pixel's relative position:

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

Correlation Filtering

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

Move v (column) first



$= (0 \times 1) + (0 \times 2) + (0 \times 3)$
$+ \quad (0 \times 4) + (0 \times 5) + (0 \times 6)$
$+ \quad (90 \times 7) + (0 \times 8) + (0 \times 9)$

$= 630 = G[2,9]$

› Steps:

1. Filter size $= (2k+1) \times (2k+1)$
   If filter size $= 3 \times 3$, then $k = 1$
2. When $i = 2, j = 9, F[2,9]$ is the center pixel
   From $u = -k = -1, v = -k = -1$
   $F[i+u, j+u] = F[2-1, 9-1] = F[1,8]$
3. Multiply with corresponding filter value
4. Repeat until $u = k = 1, v = k = 1$
   $F[i+u, j+u] = F[2+1, 9+1] = F[3,10]$
5. Sum all values and place in $G[i,j] = G[2,9]$

Correlation Filtering

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

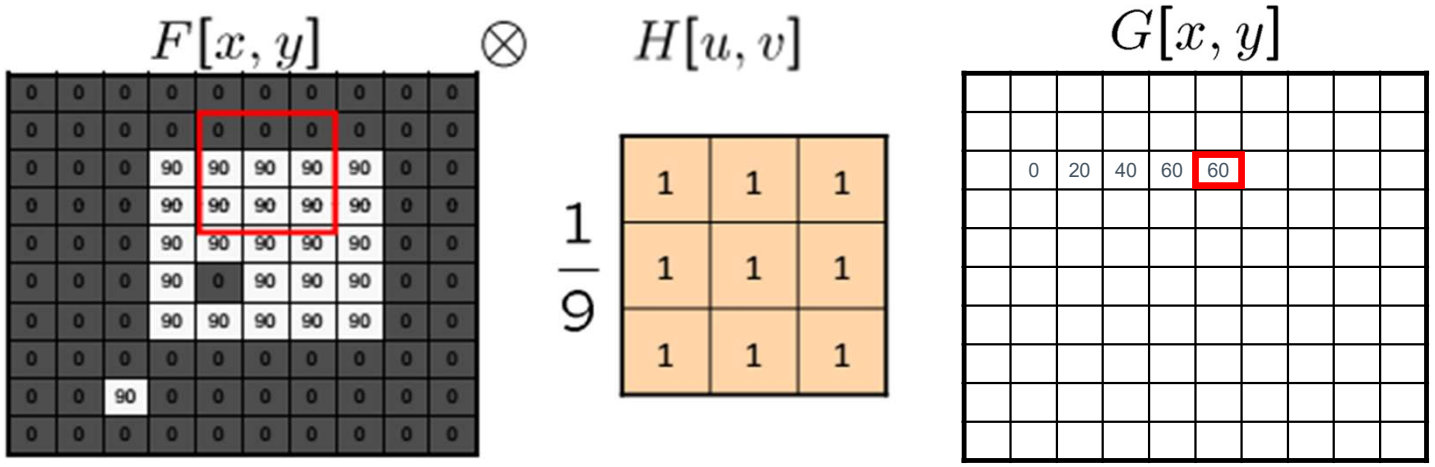
› Cross-correlation: $G = H \otimes F$

› Summary:

- Filtering an image: Replace each pixel with a linear combination of its neighbors
- The filter "kernel" or "mask" $H(u,v)$ is the prescription for the weights in the linear combination

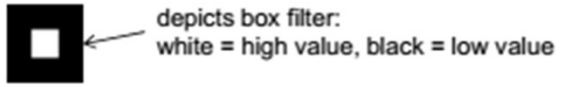
Correlation Filtering

› What values belong in the kernel H for the moving average example?

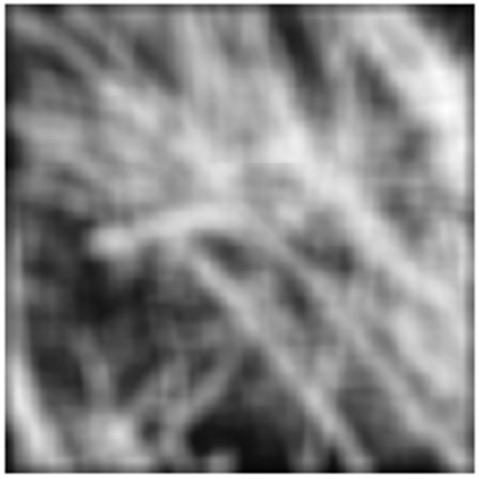


Filter #1: Moving Average

Correlation Filtering



original

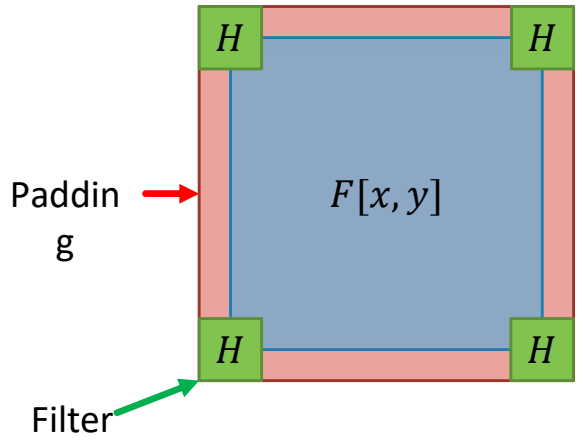


filtered

› What can we expect from the output if the filter size is 5×5 or 7×7 instead of 3×3?

Problem: Boundary issue

- What about near the edge?
  - Filter window falls off the edge of the input image, some pixels are not defined ⇒ Need to extrapolate
  - Some common padding methods:
    - Constant value (with 0's we get zero-padding)
    - Wrap around
    - Copy edge
    - `numpy.pad` has a lot more options.



Filter #2: Gaussian Filter

› What if we want the nearest neighbouring pixels to have more influence on the output?

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

$F[x, y]$

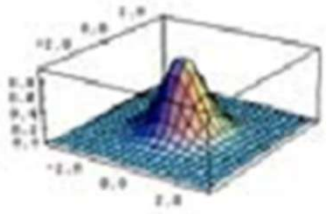
$$\frac{1}{16}$$

1	2	1
2	4	2
1	2	1

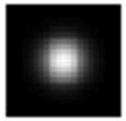
$H[u, v]$

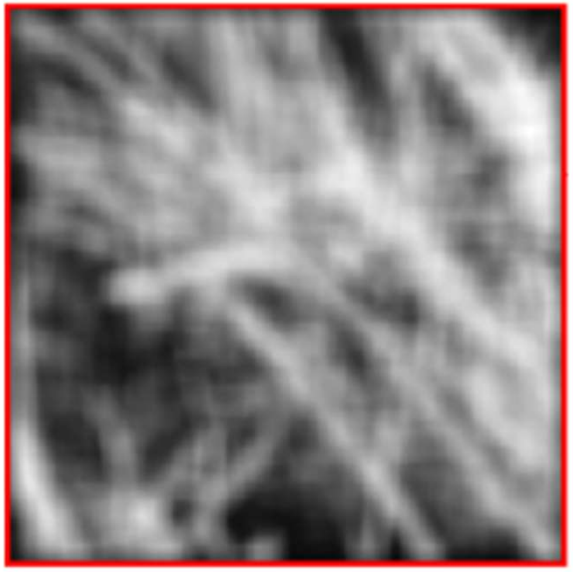
This kernel is an approximation of a 2d Gaussian function:

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



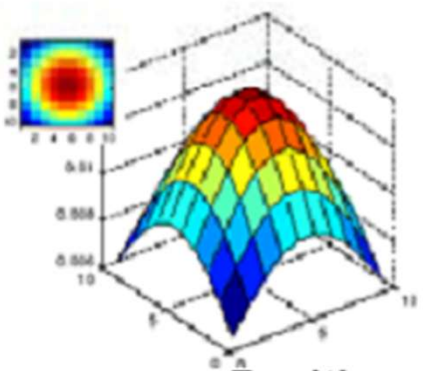
Gaussian Filter Smoothing



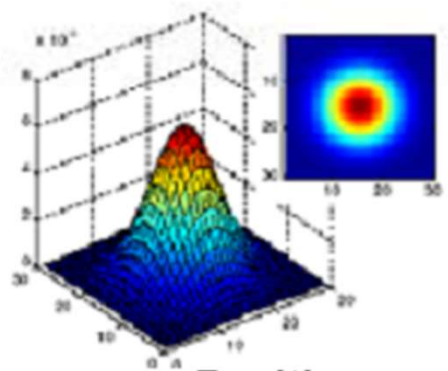


Gaussian Filter Parameters

- What parameters are important?
- **(1) Size** of kernel or mask
  - Note: Gaussian function has infinite support, but discrete filters use finite kernels



σ = 5 with 10 x 10 kernel

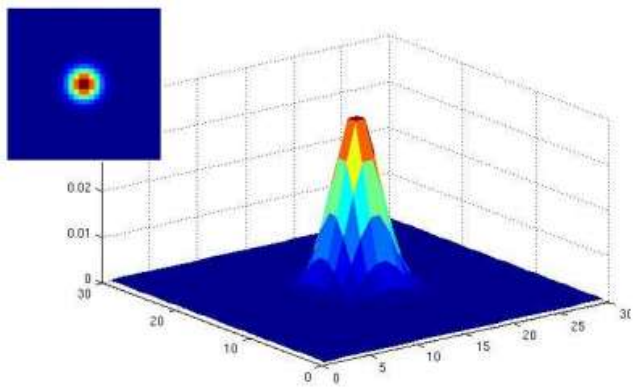


σ = 5 with 30 x 30 kernel

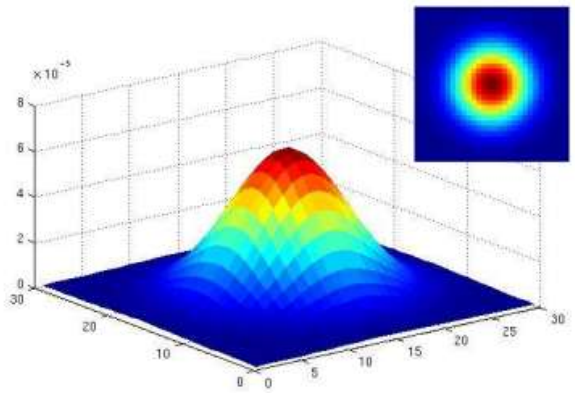
Gaussian Filter Parameters

› What parameters are important?

› **(2) Variance** of Gaussian determines extent of smoothing



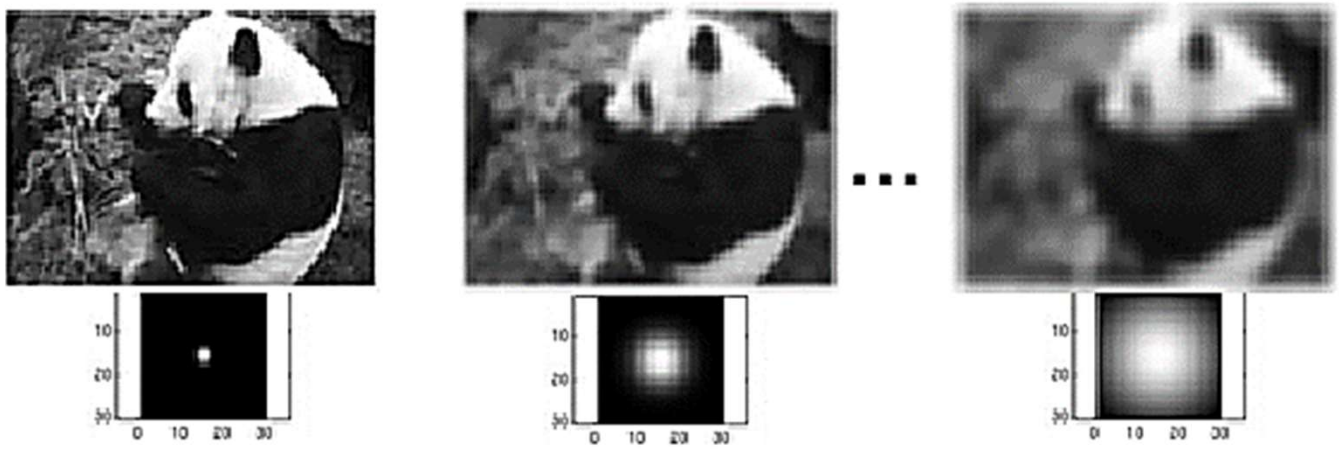
Size $= 30 \times 30$
$\sigma = 2$



Size $= 30 \times 30$
$\sigma = 5$

σ parameter

› The σ parameter is the “scale” or “width” or “spread” of the Gaussian kernel ⇒ controls the amount of smoothing

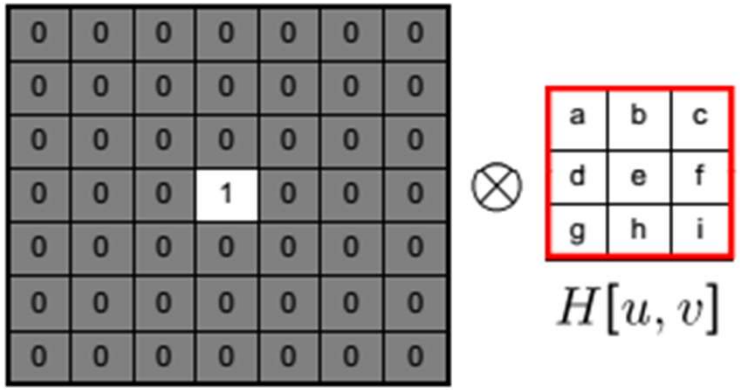


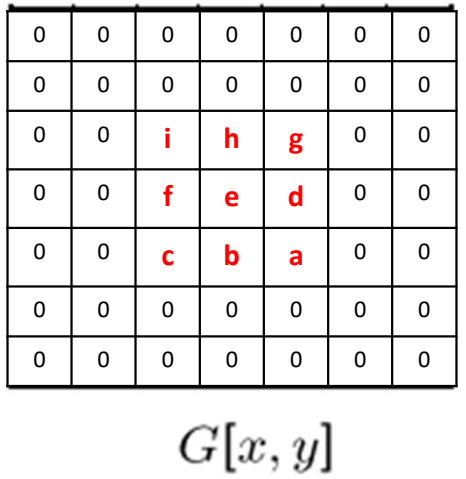
Properties of Smoothing

- Values are **positive**
- **Sum to 1** ⇒ constant regions same as input
- Amount of smoothing proportional to mask size
- “Low-pass” filtering, remove “high-frequency” components

Filtering an Impulse Signal

› What is the result of filtering the impulse signal (image) F with an arbitrary kernel H?





Convolution

› Convolution:

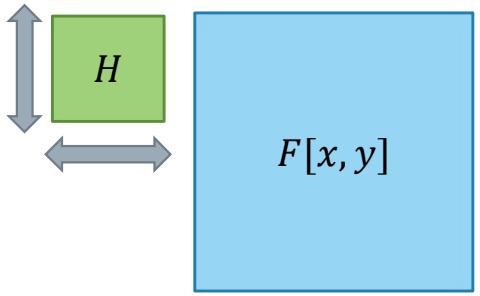
› Flip the filter in both dimensions (bottom to top, right to left)

› Then apply cross-correlation

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

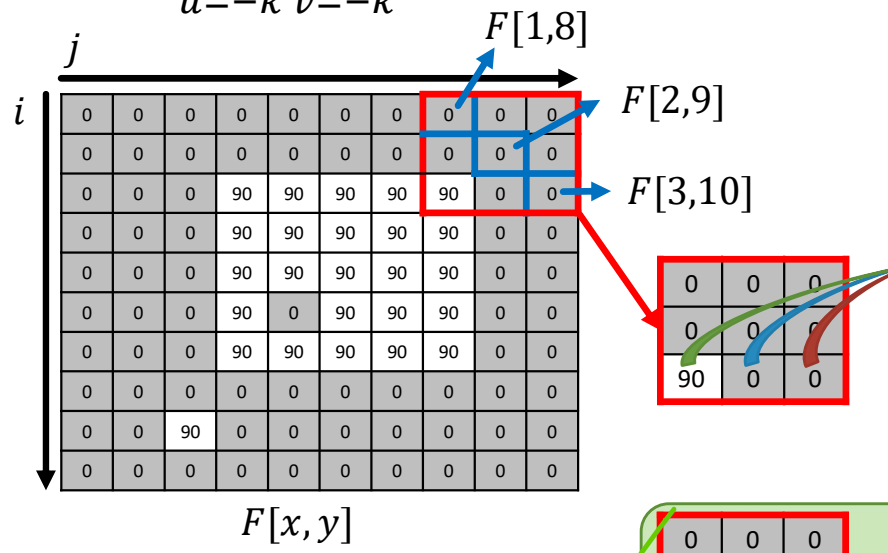
$$G = H \star F$$

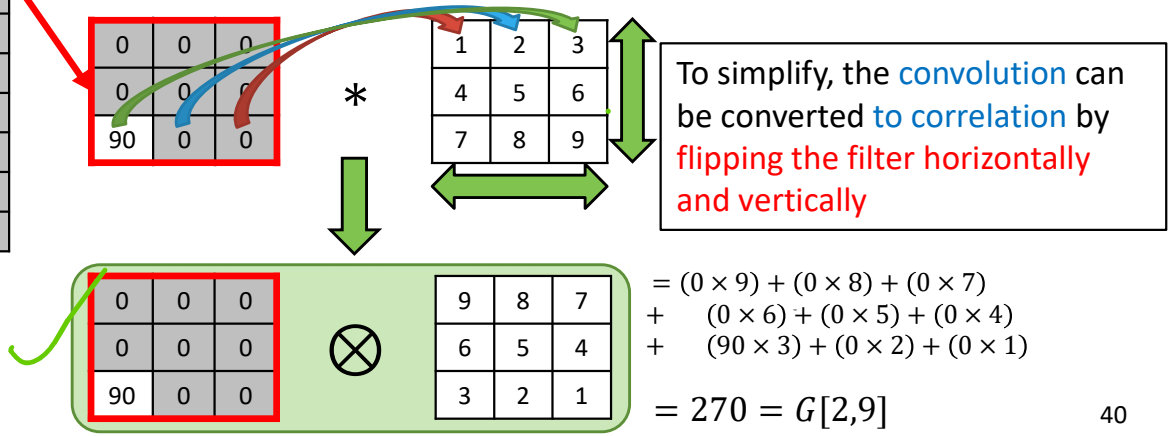
Notation for convolution operator



Convolution

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





› Steps:

1. Filter size $= (2k+1) \times (2k+1)$
   If filter size $= 3 \times 3$, then $k = 1$
2. When $i = 2, j = 9, F[2,9]$ is the center pixel
   From $u = -k = -1, v = -k = -1$
   $F[i-u, j-u] = F[2+1, 9+1] = F[3,10]$
3. Multiply with corresponding filter value
4. Repeat until $u = k = 1, v = k = 1$
   $F[i-u, j-u] = F[2+1, 9+1] = F[1,8]$
5. Sum all values and place in $G[i,j] = G[2,9]$

Convolution vs. Cross-correlation

› Convolution

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

$$G = H * F$$

› Cross-correlation

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

$$G = H \otimes F$$

Important: If the kernel is **symmetric**, **Convolution = Correlation**

Convolution is useful when dealing in the frequency domain (Fourier transform) to enable easy combination of more than 1 filter. Nice explanation on the differences: http://www.cs.umd.edu/~djacobs/CMSC426/Convolution.pdf

Separability of Filters

› In some cases, filters are **separable** ⇒ can be factored into two steps
  – Convolve all rows
  – Convolve all columns

Separability of the Gaussian filter

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

2D convolution (center location only)

$$\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix} * \begin{bmatrix}2&3&3\\3&5&5\\4&4&6\end{bmatrix}$$

The filter factors into a product of 1D filters:

$$\begin{bmatrix}1&2&1\\2&4&2\\1&2&1\end{bmatrix} = \begin{bmatrix}1\\2\\1\end{bmatrix} \times \begin{bmatrix}1&2&1\end{bmatrix}$$

Perform convolution along rows:

$$\begin{bmatrix}1&2&1\end{bmatrix} * \begin{bmatrix}2&3&3\\3&5&5\\4&4&6\end{bmatrix} = \begin{bmatrix}&11&\\&18&\\&18&\end{bmatrix}$$

Followed by convolution along the remaining column:

$$\begin{bmatrix}1\\2\\1\end{bmatrix} * \begin{bmatrix}&11&\\&18&\\&18&\end{bmatrix} = \begin{bmatrix}&&\\&65&\\&&\end{bmatrix}$$

Separability of Filters

› What is the complexity of filtering an $n \times n$ image with an $m \times m$ kernel?

– $O(n^2m^2)$

1	2	1
2	4	2
1	2	1

$*$

2	3	3
3	5	5
4	4	6

› What if the kernel is separable?

– $O(n^2m) = O(2n^2m)$

1	2	1

$*$

2	3	3
3	5	5
4	4	6

$=$

	11	
	18	
	18	

1
2
1

$*$

	11	
	18	
	18	

$=$

	65	

Separability of Filters

› Is this separable? If yes, what is the separable version?

$$\frac{1}{K^2}\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & 1 & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

$$\frac{1}{K}\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}$$

› What does this filter do?

Separability of Filters

› Is this separable? If yes, what is the separable version?

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

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

› What does this filter do?

Separability of Filters

› Is this separable? If yes, what is the separable version?

$$\frac{1}{8}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

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

› What does this filter do?

Separability of Filters

› Is this separable? If yes, what is the separable version?

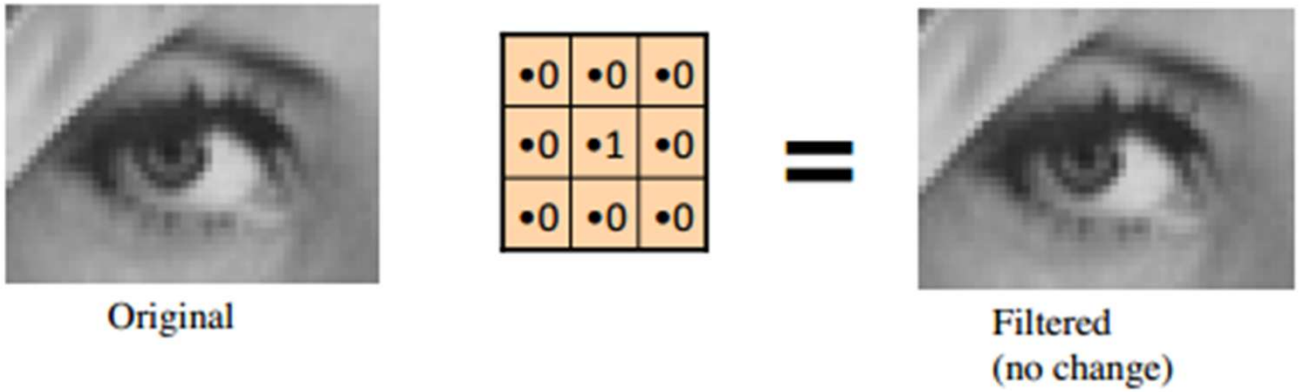
$$\frac{1}{4}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

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

› What does this filter do?

Image Sharpening

Intuition of Filtering



Intuition of Filtering

Shifted left
by 1 pixel

Original

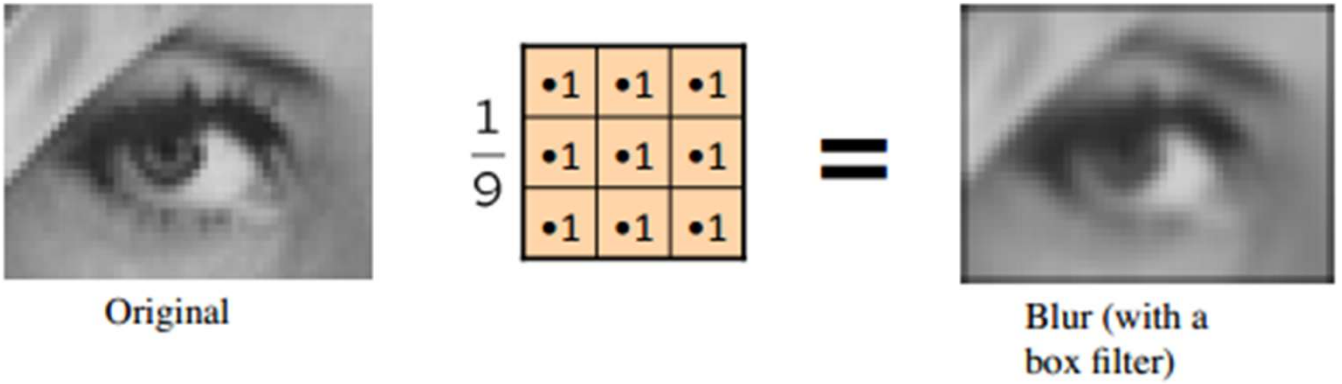
•0	•0	•0
•0	•0	•1
•0	•0	•0

=

?

Shifted right
by 1 pixel

Intuition of Filtering



Original

Blur (with a box filter)

Sharpening

› **Sharpening by filtering**: Accentuates differences with local average

Original

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

-1	-1	-1
-1	17	-1
-1	-1	-1

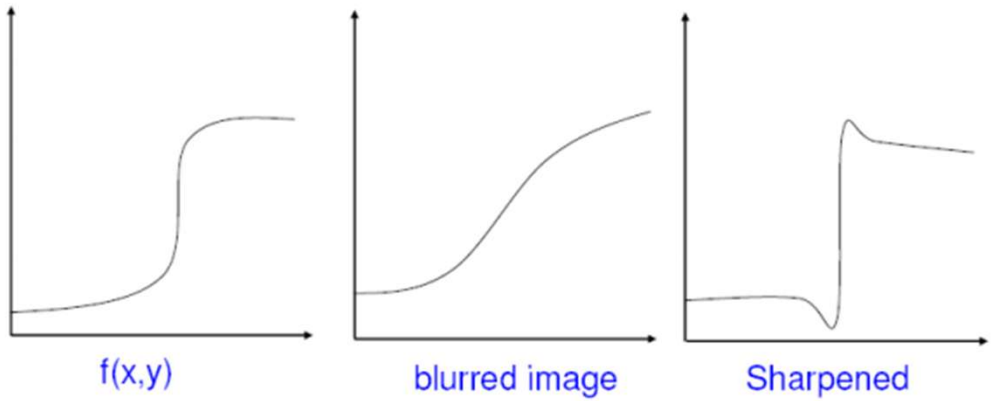
$=$

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

› This is also related to a process called **Unsharp Masking**

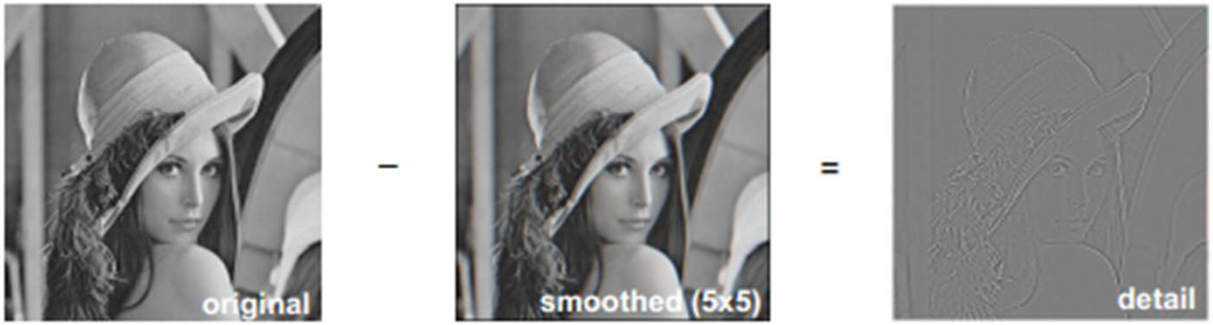
Unsharp Masking

- A process used many years in the publishing industry to sharpen images
- **Details = Original image – Smoothened image**
- **Sharpened image = Original image + Details**

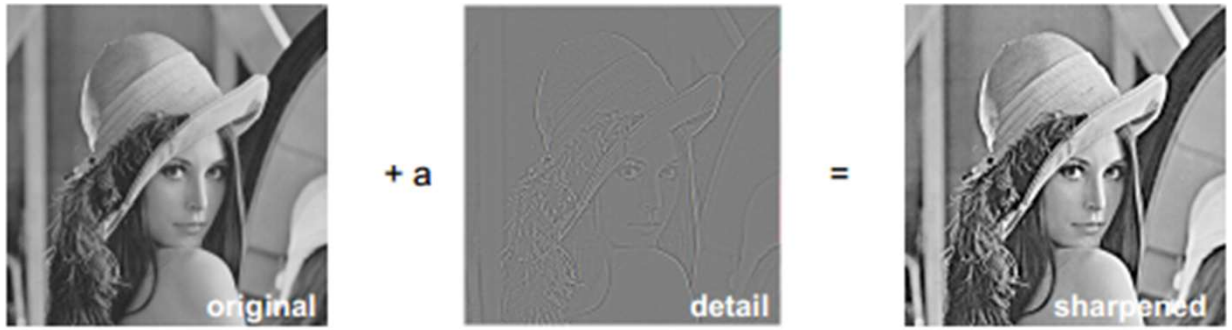


Unsharp Masking

› What does blurring take away?



› Adding the details back…



Sharpening Filter

Original

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

(Note that filter sums to 1)

"details of the image"

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

Amount of Sharpening

- **Details = Original image – Smoothened image**
- **Sharpened image = Original image + Details**
  - How can we control the amount of sharpening that is applied?

The strength of the details/smoothing filter!

Non-linear Filters

Non-linear Filtering

› So far, we look at linear filtering

Linear combination of multiplied terms

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

$$G = H * F$$

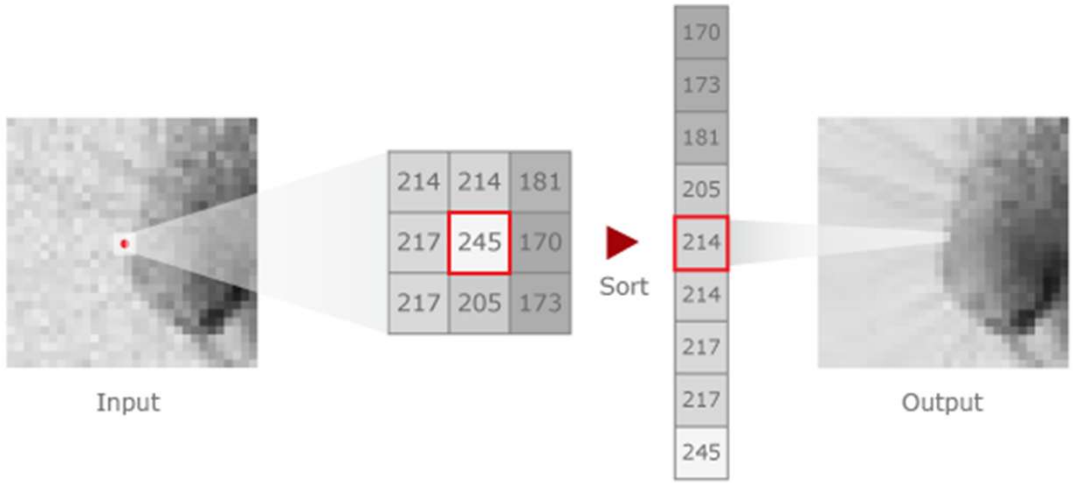
› What about **non-linear filtering**?

› Can the choice of filter $H[u,v]$ produce non-linear filtering?

Median Filter

› A type of **order-statistics** filter – “statistical estimator”

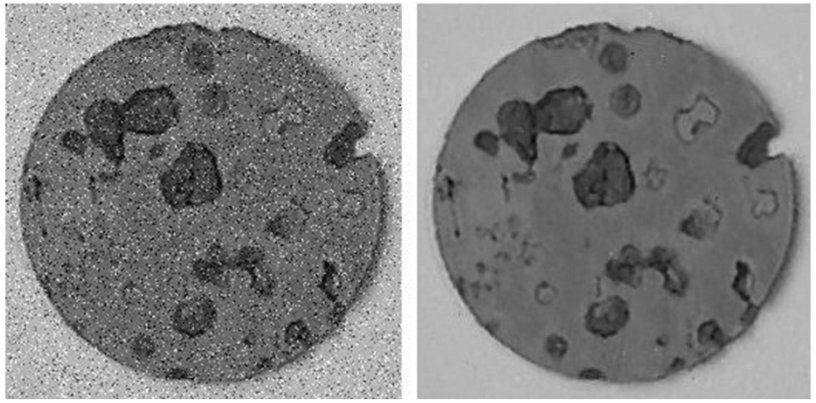
› No new pixel values are introduced



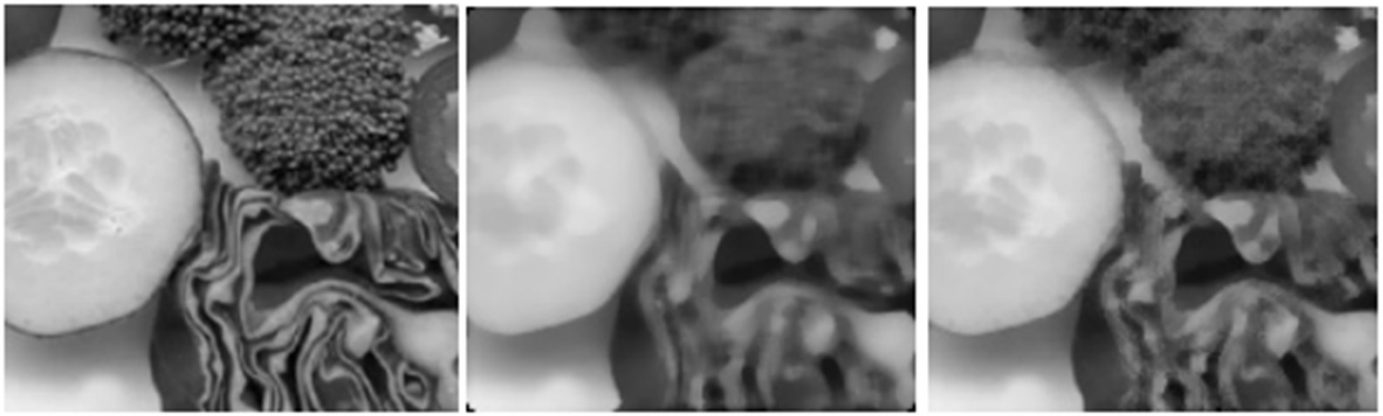
› **Removes spikes**: Good for impulse, salt & pepper noise

Median Filter

› Stray white pixels (very high values) or stray black pixels (very low values) can be dealt with



Median Filter



› Example:
  - Applying 7 × 7 median filter (middle figure): broccoli loses details
  - Applying 7 × 7 multi-stage median filter: less smoothing occurs, some details are maintained

Median Filter

› **Multi-stage** median filter:

– Median of a set of different median filters (obtained in different neighbourhoods)

$$
\begin{aligned}
y_{ij} &= med(z_1, z_2, z_3, z_4) \\
z_1 &= med(\{x_{uv} | x_{uv} \in N_{ij}^1\}) \\
z_2 &= med(\{x_{uv} | x_{uv} \in N_{ij}^2\}) \\
z_3 &= med(\{x_{uv} | x_{uv} \in N_{ij}^3\}) \\
z_4 &= med(\{x_{uv} | x_{uv} \in N_{ij}^4\})
\end{aligned}
$$

– Preserves sharp corners

Median Filter

› Drawback: Tend to be better at rejecting outliers but not so good at handling noise without outliers (e.g. Gaussian white noise)

› Averaging filter / weighted average filter

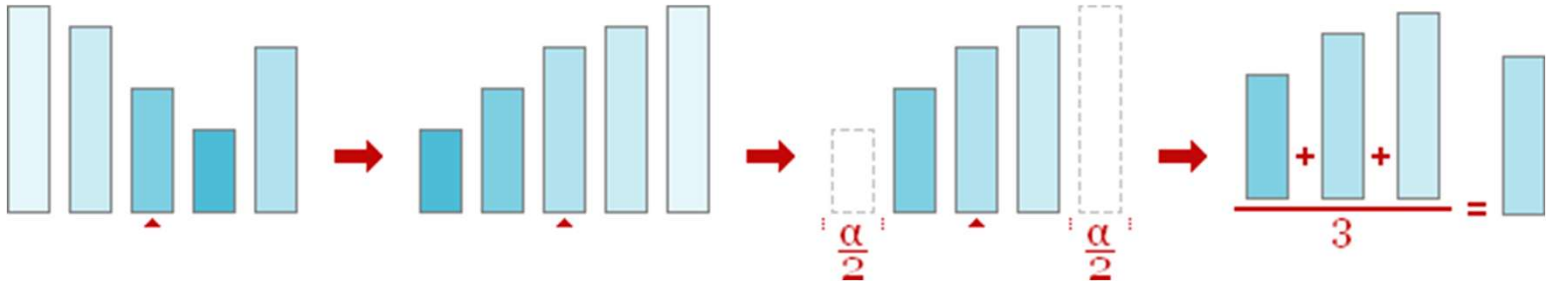
- too much blurriness, noise remains but smoothened

Alpha-trimmed Mean Filter

- Select the average of the values within a window which excludes a percent of the largest and smallest values in the neighbourhood
- Delete $\alpha/2$ lowest and $\alpha/2$ highest values in the neighbourhood S_{XY}, find average of remaining pixels:

$$\hat{f}(x,y) = \frac{1}{mn-\alpha} \sum_{(s,t)\in S_{XY}} g_r(s,t)$$

Alpha-trimmed Mean Filter



› How should α be chosen?

SUMMARY

- Image Filtering
  - Neighbourhood Processing, Motivation, Masking, Moving Average
- Correlation and Convolution
  - Box filter, Gaussian Filter, Smoothing, Filter Separability
- Image Sharpening
  - Unsharp Masking, Sharpening Filter
- Non-linear filters
  - Median Filter, Alpha-trimmed Mean Filter
- Next:
  - Edge Detection
- Recommended Reading
  - [Gonzalez&Woods] Chapter 3
  - [Forsyth & Ponce] Chapter 8 & 10