

# Image Filtering


Fundamentals of Data Science '24

*Luca Scofano*




SAPIENZA  
UNIVERSITÀ DI ROMA

# Notes

- This portion of the programme will be part of the Homework
-  Open Ended questions

# Table of Content

- Definition of an image 
- Definition of a filter
- Convolutions
- Examples of Filters
- Multi-Scale Image Representation

An image is a **function** that  
maps **locations** to **pixels**

$$f(x, y) = \text{pixel}$$

This is an example using an  
**RGB** image

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

$[0, 255]$  which equals to  $2^8$

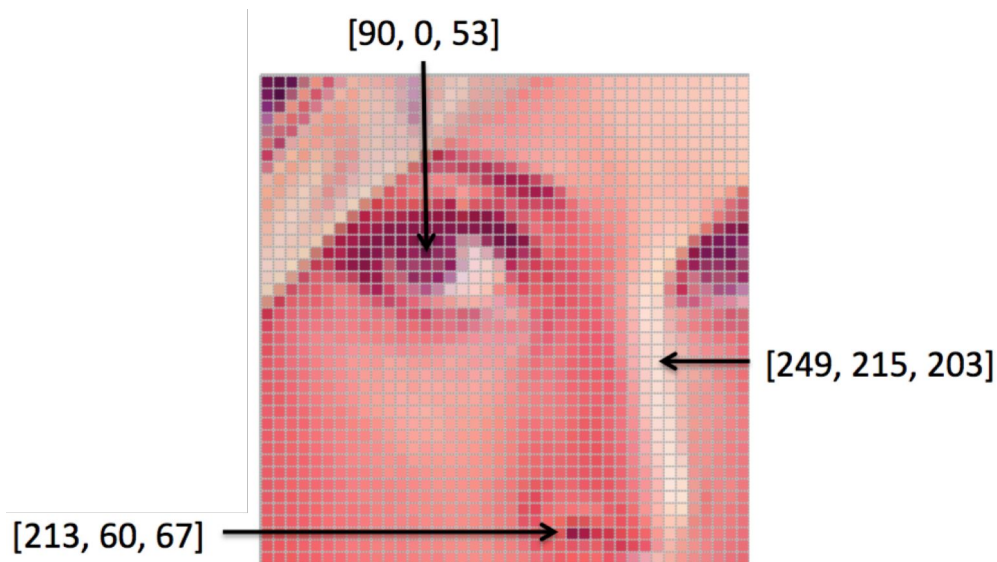
What is the number of  
**combinations** a single pixel  
can have?

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

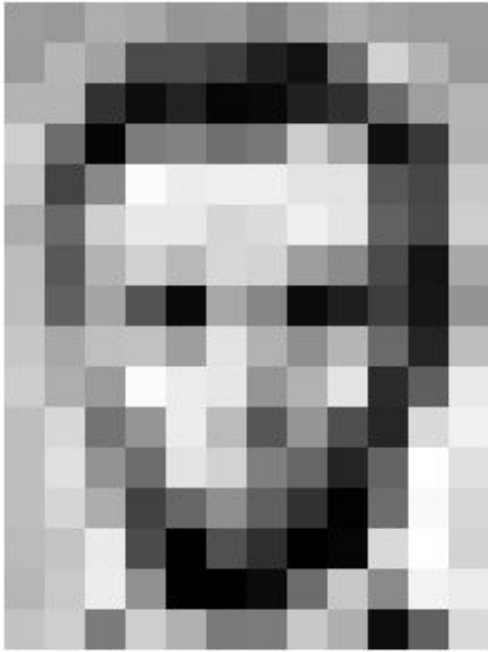
$$256 * 256 * 256 = 16,777,216$$



What is the number of **combinations** a single pixel can have?

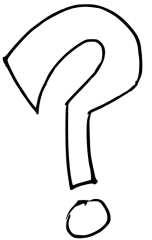


# A more intuitive example (**Greyscale**)



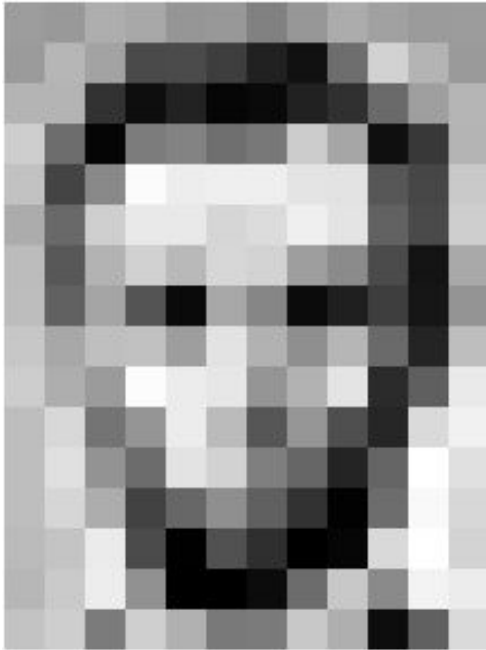
What are the values a single pixel can have?

**256**





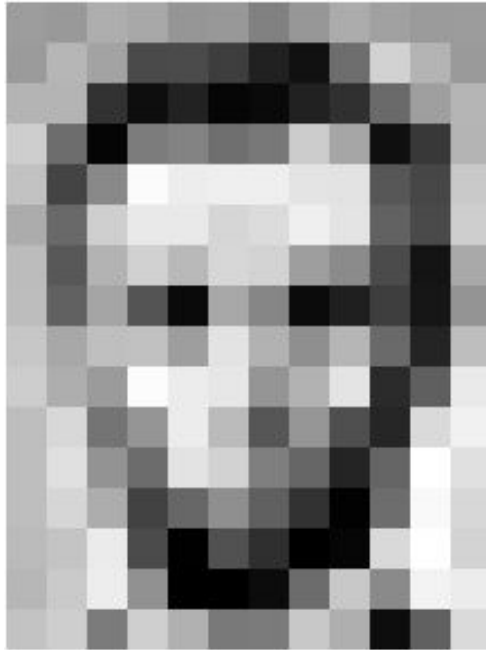
# A more intuitive example (**Greyscale**)



=

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

# A more intuitive example (**Greyscale**)



157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	106	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	106	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	85	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	96	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

# Table of Content

- Definition of an image
- Definition of a filter
- Convolutions
- Examples of Filters
- Multi-Scale Image Representation

Why do we need  
Image Filtering?



Image **Deblurring**

# Why do we need Image Filtering?



# How do we define a **filter**?

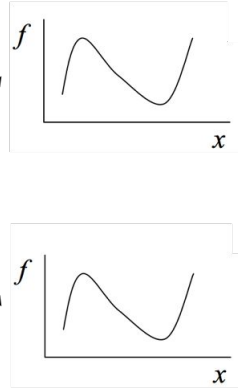
$$f(x, y)$$

An **image** is a **function** that maps **locations** to **pixels**

$$h(f(x, y))$$

A **filter** is a **function** applied to the **pixels** of an image

# Beware

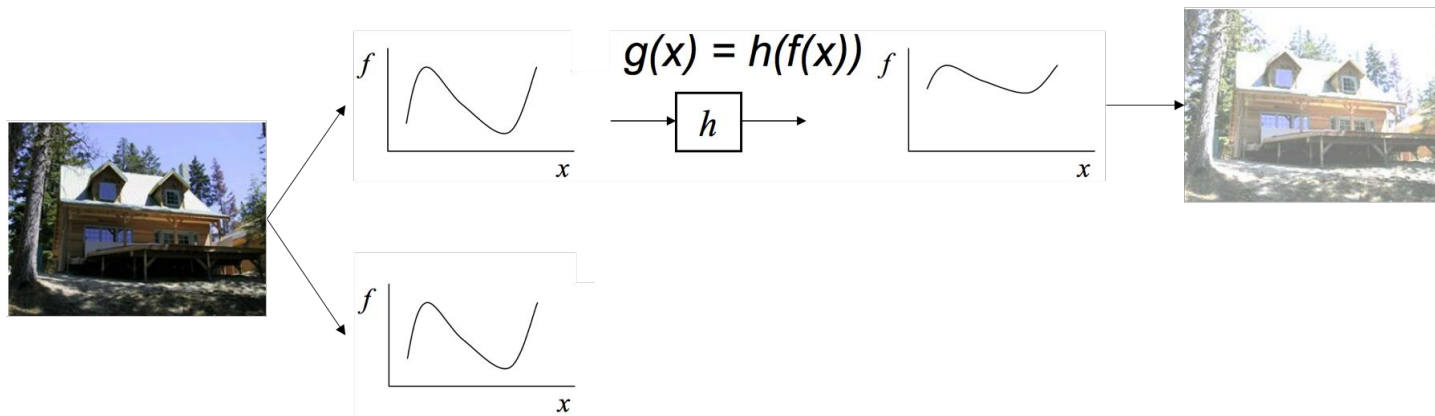


We represent the image as a continuous **signal**

# Beware!

**Keeps the same  
positions** of the pixels

Image Filtering





# Beware!

**Keeps the same positions of the pixels**

Image Filtering

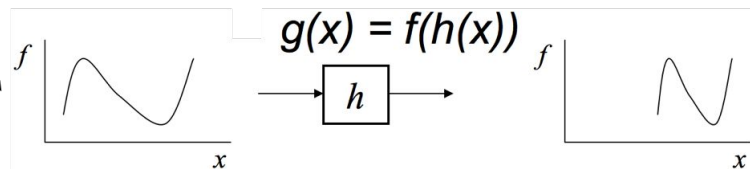
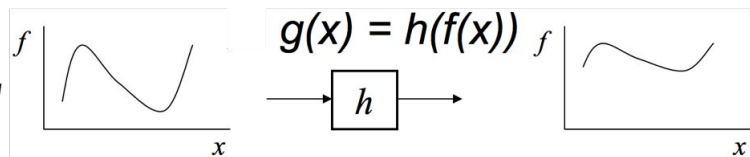


Image Warping

**Changes the positions of the pixels**

# Why do we need Image Filtering?

- Filtering for enhancement **improve contrast**
- Filtering for smoothing **removes noise**
- Filtering for **template matching** detects known patterns

# Table of Content

- Definition of an image
- Definition of a filter
- **Convolutions**
- Examples of Filters
- Multi-Scale Image Representation

# Convolutions

- **Feature Extraction:** Convolutions enable the detection of local patterns. In deep learning, convolutional layers automatically learn these features at different levels of abstraction.
- **Localized Operations:** Convolutions operate on small, localized regions (called kernels or filters) of an input image or signal, making them efficient for tasks like smoothing, sharpening, or detecting gradients.
- **Shift Invariance:** Convolutional operations preserve the spatial structure of the input data, allowing the model to detect features regardless of their location within the image. This is crucial for tasks like object recognition.

# 2D-Convolutions

Which mathematical operations do we perform?

0	-1	0
-1	5	-1
0	-1	0

Kernel

1	3	2	0	1
2	1	0	1	3
1	3	2	1	1
2	0	2	3	2
1	3	1	2	1

Input Image



# 2D-Convolutions

## Element-Wise Matrix Multiplication

+

## Summation

0	-1	0
-1	5	-1
0	-1	0

Kernel

1	3	2	0	1
2	1	0	1	3
1	3	2	1	1
2	0	2	3	2
1	3	1	2	1

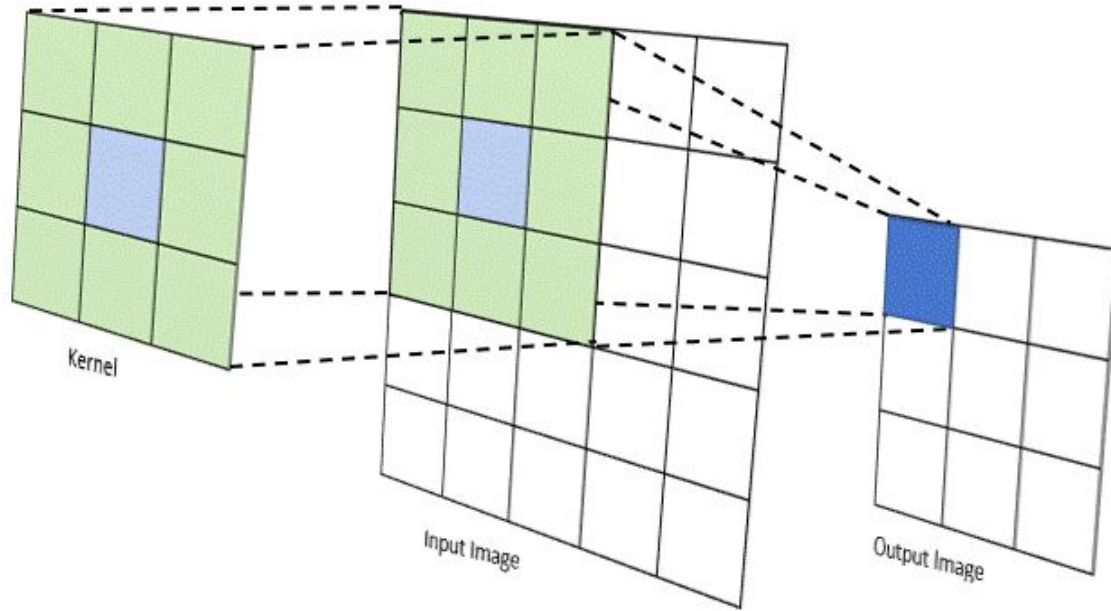
Input Image

$$\begin{aligned} & (0)(1) + (-1)(3) + (0)(2) + \\ & (-1)(2) + (5)(1) + (-1)(0) + \\ & (0)(1) + (-1)(3) + (0)(2) \\ & = -3 \end{aligned}$$

-3		

Output Image

# 2D-Convolutions



# Some Notation

$$y(n, m) = \sum_k \sum_l x(k, l) k(n - k, m - l)$$

0	-1	0
-1	5	-1
0	-1	0

Kernel

1	3	2	0	1
2	1	0	1	3
1	3	2	1	1
2	0	2	3	2
1	3	1	2	1

Input Image

$(0)(1) + (-1)(3) + (0)(2) +$ $(-1)(2) + (5)(1) + (-1)(0) +$ $(0)(1) + (-1)(3) + (0)(2)$ $= -3$		

Output Image



# Beware

$$y(n, m) = \sum_k \sum_l x(k, l) \underline{k(n - k, m - l))}$$

If the kernel were not **flipped**, the operation would correspond to **cross-correlation**, which is slightly different from convolution. Cross-correlation measures **similarity between two signals**.

# 2D-Convolutions

What if we want to keep the **same dimensions in output**?

0	-1	0
-1	5	-1
0	-1	0

Kernel

1	3	2	0	1
2	1	0	1	3
1	3	2	1	1
2	0	2	3	2
1	3	1	2	1

Input Image



# 2D-Convolutions + Padding

0	-1	0
-1	5	-1
0	-1	0

Kernel

1	1	3	2	0	1	1
1	1	3	2	0	1	1
2	2	1	0	1	3	3
1	1	3	2	1	1	1
2	2	0	2	3	2	2
1	1	3	1	2	1	1
1	1	3	1	2	1	1

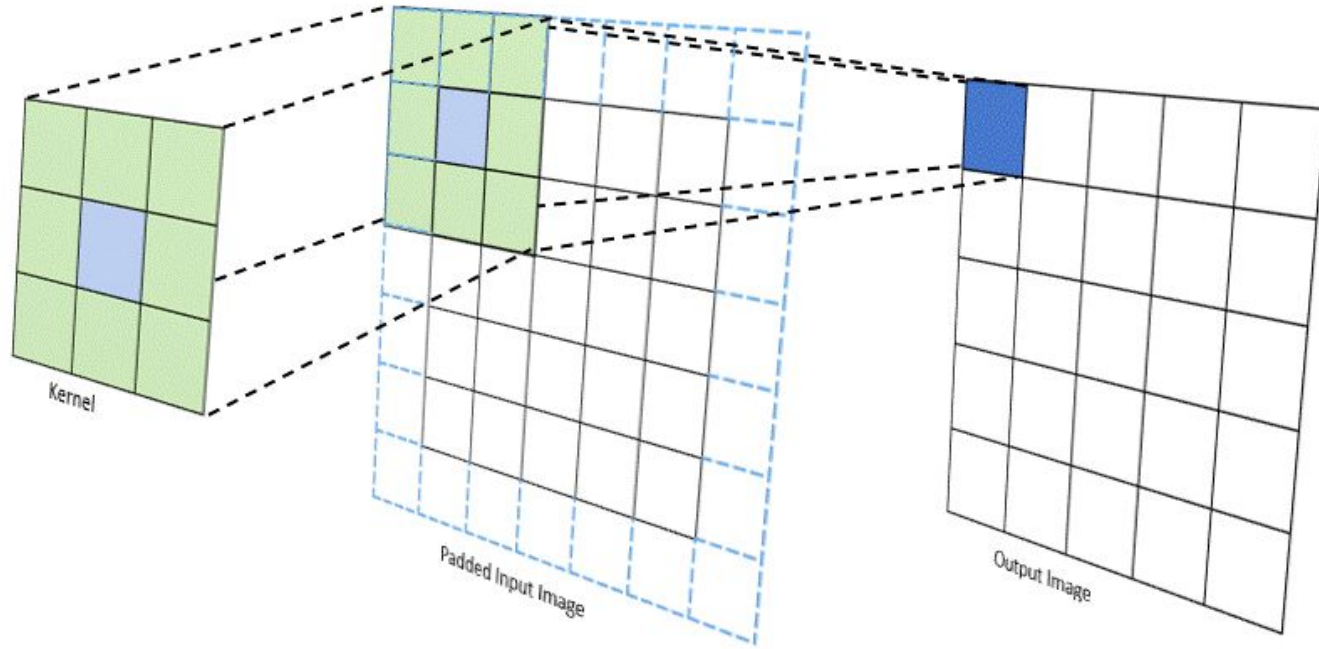
Padded Input Image

$$\begin{aligned} &(0)(1) + (-1)(1) + (0)(3) + \\ &(-1)(1) + (5)(1) + (-1)(3) + \\ &(0)(2) + (-1)(2) + (0)(1) \\ &= -2 \end{aligned}$$

-2				




Output Image

# 2D-Convolutions + Padding



# Table of Content

- Definition of an image
- Definition of a filter
- Convolutions
- **Examples of Filters**
- Multi-Scale Image Representation

Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ 
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$ 
Mean Blur	$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$ 

**The identity kernel** (a 3x3 matrix with 1 in the center and 0 elsewhere) **leaves the image unchanged**. It simply multiplies each pixel by 1 and adds 0 to its neighbors, preserving the original image.

**The sharpen kernel** enhances edges by amplifying the difference between a pixel and its neighbors. The center value (5) boosts the current pixel, while the negative values around it subtract the neighbors.

**The mean blur kernel** averages each pixel with its 8 neighbors equally (each weighted 1/9). This smooths out differences between adjacent pixels, resulting in a blurred image where fine details are reduced.



# Variation of the Mean blur

Gaussian Blur

$$\begin{bmatrix} 1/16 & 2/16 & 1/16 \\ 2/16 & 4/16 & 2/16 \\ 1/16 & 2/16 & 1/16 \end{bmatrix}$$



**Similar to mean blur**, but weights pixels based on their distance from the center using a Gaussian distribution. This creates a more natural-looking blur, as it mimics the way our eyes perceive out-of-focus areas.

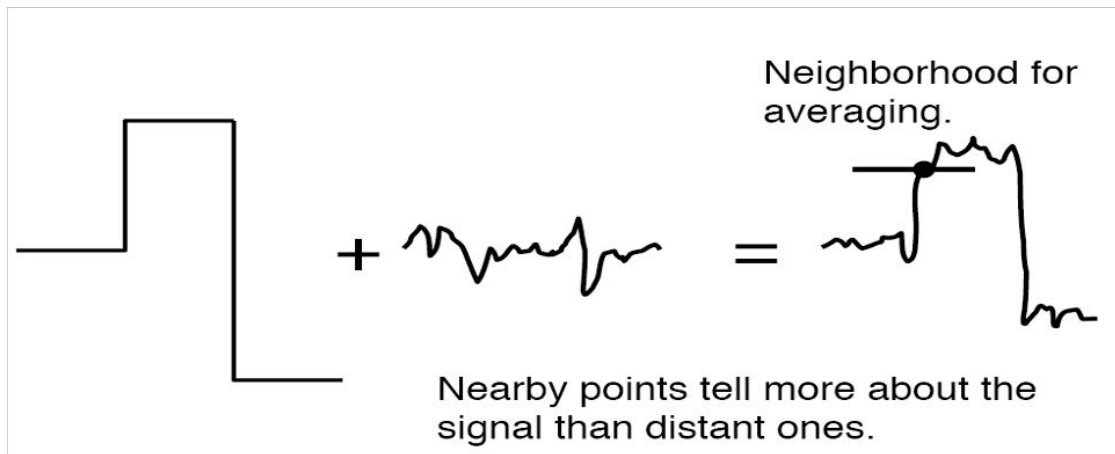
**Which filter do we want to  
use to reduce noise in an  
image?**





- Image  $I$  = Signal  $S$  + Noise  $N$ :

$$S + N = I$$



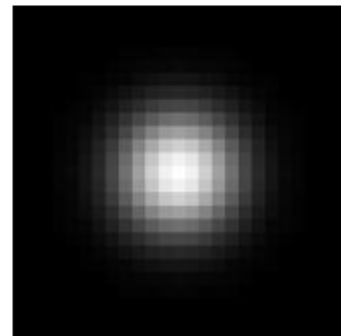
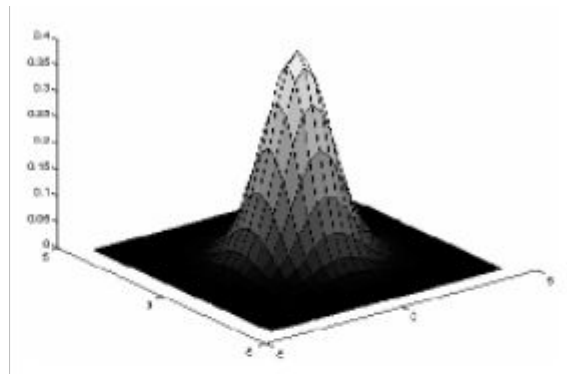
- Image  $I = \text{Signal } S + \text{Noise } N$ 
  - i.e. noise does not depend on the signal

- we consider:
  - $I_i$  : intensity of  $i$ 'th pixel
  - $I_i = s_i + n_i$  with  $E(n_i) = 0$ 
    - $s_i$  deterministic
    - $n_i, n_j$  independent for  $i \neq j$
    - $n_i, n_j$  i.i.d. (independent, identically distributed)

- therefore:
  - intuition: averaging noise reduces its effect
  - better: smoothing as inference about the signal

# Proprieties

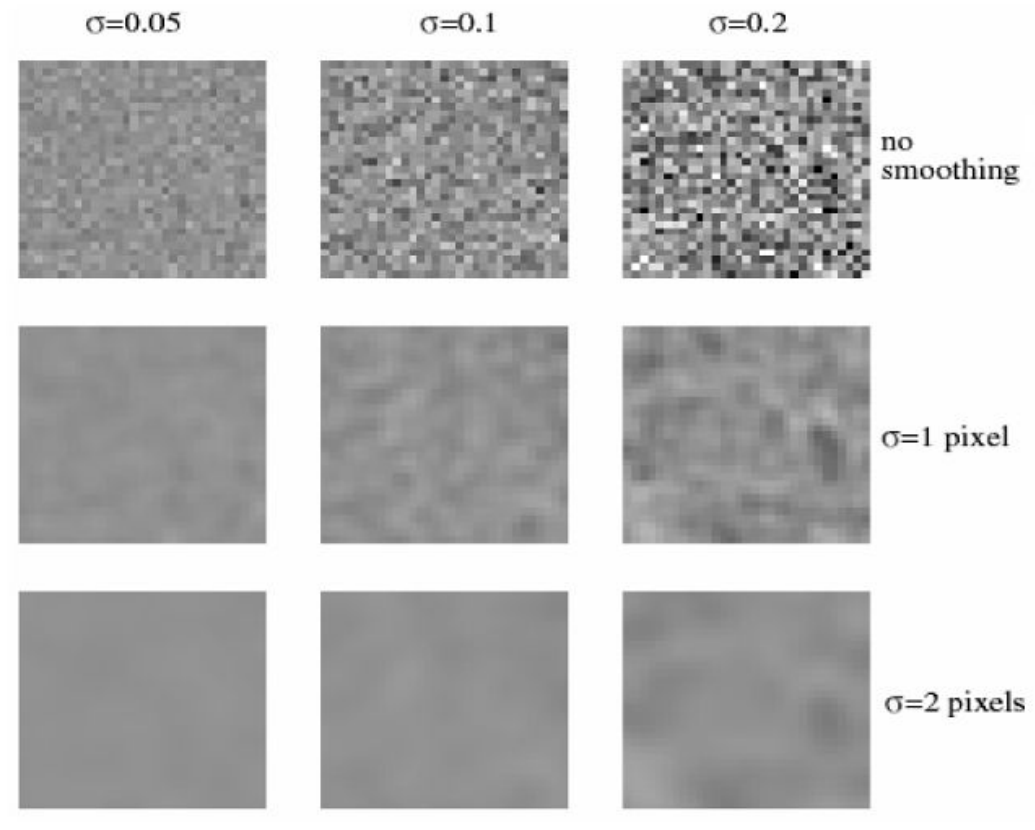
- Rotational Symmetric
- Weights nearby pixels more than distant ones



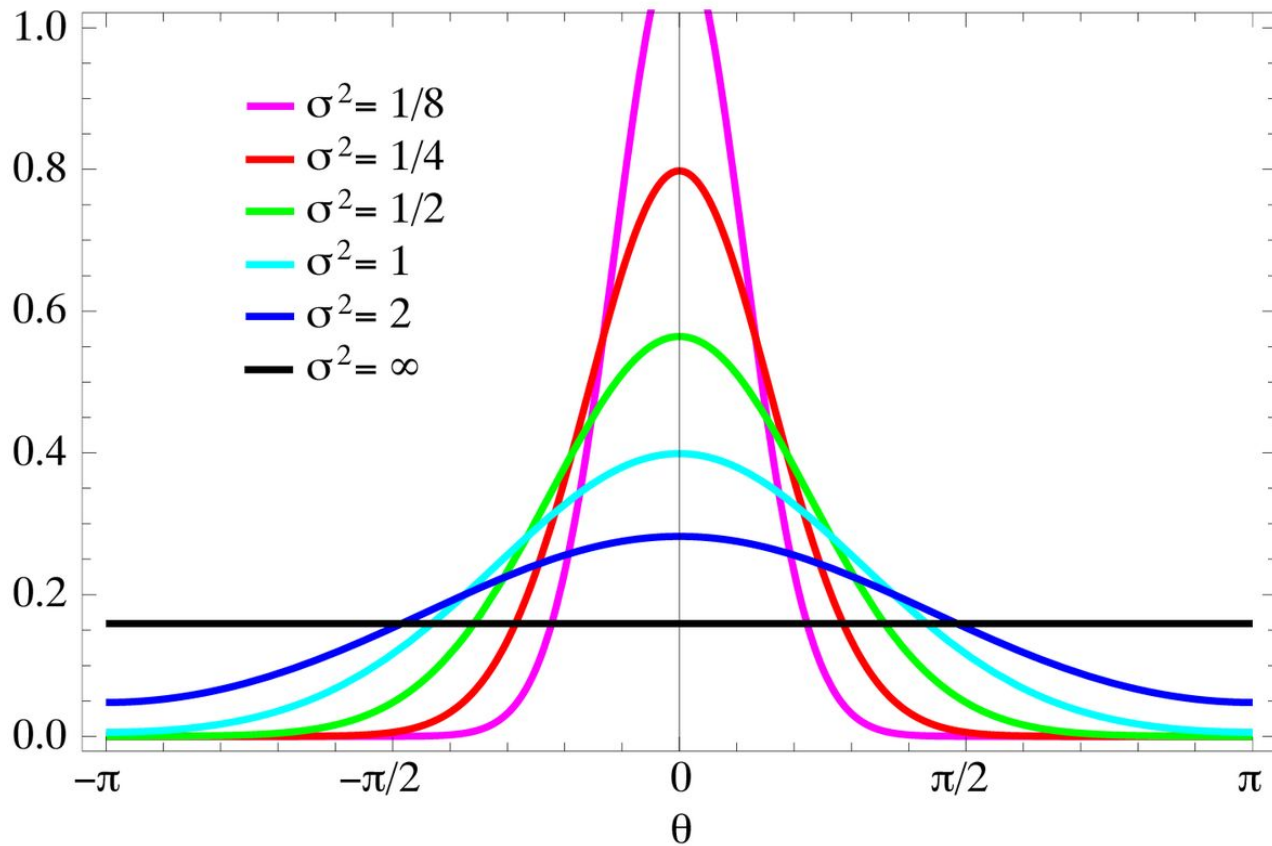
$$g(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

## Effects of smoothing:

- **each column** shows realizations of an image of Gaussian noise
- **each row** shows smoothing with Gaussians of different width



# Different $\sigma^2$



# Examples of Filters

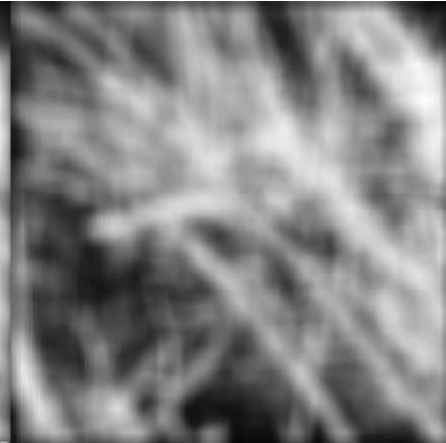
**Original**



**Gaussian**



**Mean or Box**



# Separable property

- first convolve each row with a 1D filter
- then convolve each column with a 1D filter

$$f_x \otimes f_y = f_x \otimes f_y$$

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

$$=$$

$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$
---------------	---------------	---------------

$$\otimes$$

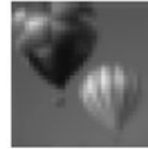
$\frac{1}{3}$
$\frac{1}{3}$
$\frac{1}{3}$

# Table of Content

- Definition of an image
- Definition of a filter
- Convolutions
- Examples of Filters
- **Multi-Scale Image Representation**

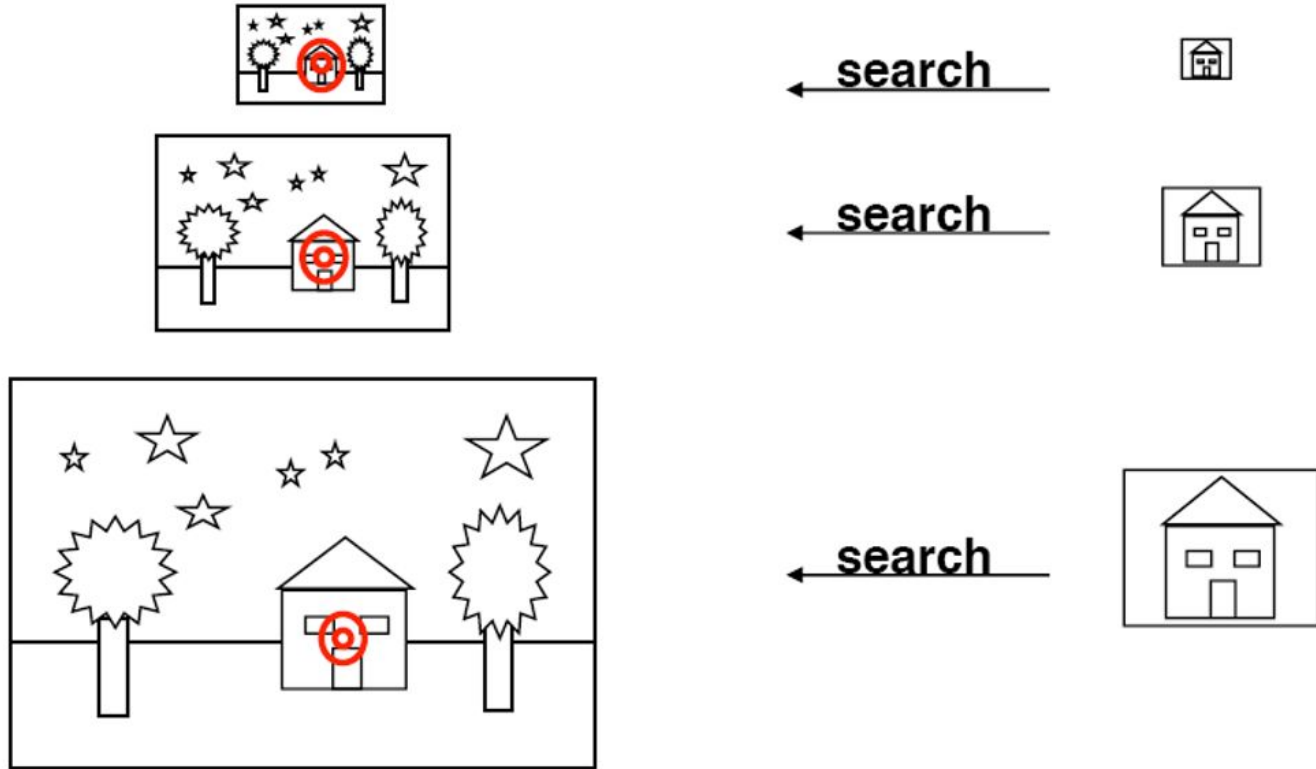


# Multi Scale



High resolution  Low resolution

# Motivation



**A specific case of  
Multi-Scale Algorithm:  
*Gaussian Pyramids***

# Algorithm

1. **Start with the original image** (base of the pyramid).

# Algorithm

1. Start with the original image (base of the pyramid).
2. Apply a **Gaussian blur to the image**. This smooths out high-frequency details.

# Algorithm

1. Start with the original image (base of the pyramid).
2. Apply a Gaussian blur to the image. This smooths out high-frequency details.
3. **Downsample** the blurred image by removing every other pixel in both horizontal and vertical directions. This reduces the image size by a factor of 4.

# Algorithm

1. Start with the original image (base of the pyramid).
2. Apply a Gaussian blur to the image. This smooths out high-frequency details.
3. Downsample the blurred image by removing every other pixel in both horizontal and vertical directions. This reduces the image size by a factor of 4.
4. **Repeat steps 2 and 3** on the resulting smaller image.

# Algorithm

1. Start with the original image (base of the pyramid).
2. Apply a Gaussian blur to the image. This smooths out high-frequency details.
3. Downsample the blurred image by removing every other pixel in both horizontal and vertical directions. This reduces the image size by a factor of 4.
4. Repeat steps 2 and 3 on the resulting smaller image.
5. Continue this process until you reach the desired number of levels or until the image becomes too small to process further.



# What happens if we do not blur the image?



Aliasing creates **artifacts**

# Problem

Aliasing occurs when high-frequency components of an image are undersampled, leading to distortions in the downsampled image.

# Solution

Gaussian blur acts as a low-pass filter. It smooths out high-frequency details in the image before downsampling. This is critical because:

- High-frequency information can't be accurately represented at lower resolutions.
- Without blurring, sharp edges and fine details could create jagged artifacts when downsampled.

# References

- <https://ai.stanford.edu/~syYeung/cvweb/tutorial1.html#:~:text=Image%20filtering%20changes%20the%20range,points%20without%20changing%20the%20colors>
- <https://medium.com/analytics-vidhya/a-beginners-guide-to-computer-vision-part-4-pyramid-3640edefb00>
- [https://medium.com/@timothy\\_terati/image-convolution-filtering-a54dce7c786b](https://medium.com/@timothy_terati/image-convolution-filtering-a54dce7c786b)
- [https://docs.opencv.org/4.x/dc/dff/tutorial\\_py\\_pyramids.html](https://docs.opencv.org/4.x/dc/dff/tutorial_py_pyramids.html)