Lecture 11: Understanding and applying basic image filtering techniques

Outlines:

- Noise in Images
- Image Filtering
- Mathematics Behind Image Filtering
- Basic Filters
- Conclusion

Applications of Filters

Image denoising,

Remove blur from an image

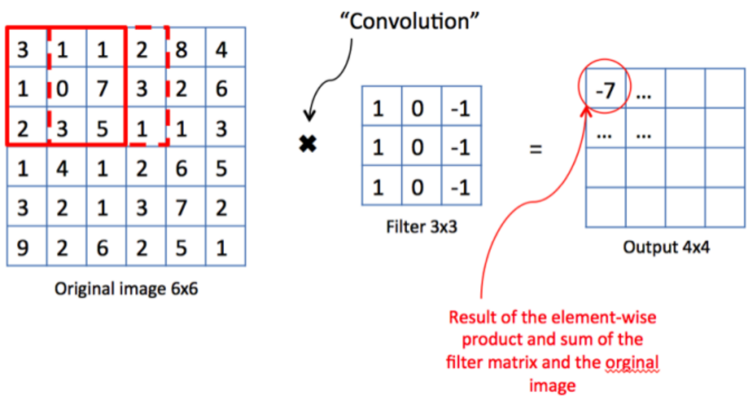
Image enhancement

Image Manipulation (get a desired result)

Convolution

Convolution is a set of mathematical operations to calculate output of a filter

- In this figure you can clearly see the process of convolution in a matrix of 6x6 by a filter of size 3x3



- In order to carry out an image filtering process, mostly convolution or correlation is performed.

Correlation

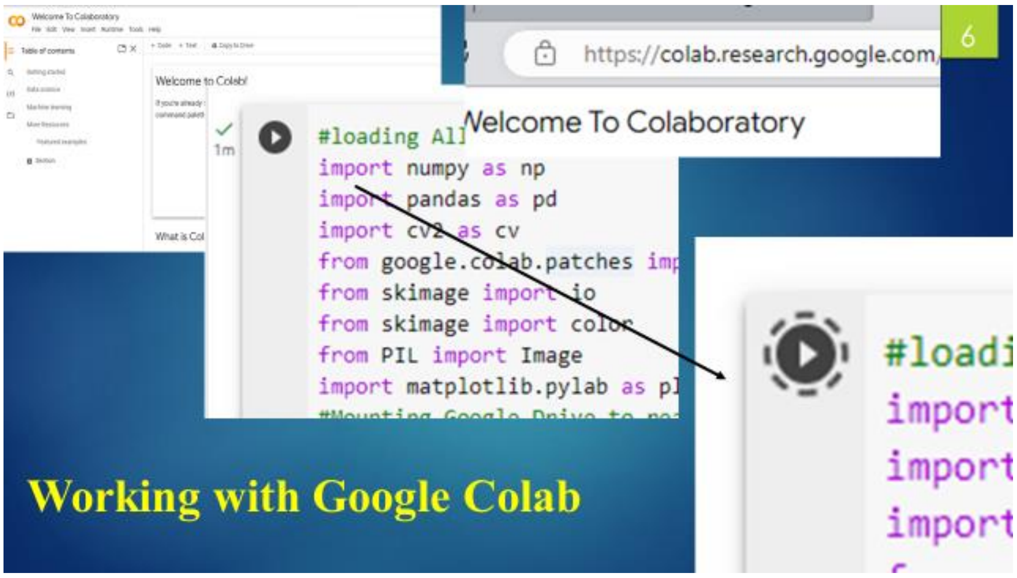
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$
$$= H \otimes F$$

Convolution

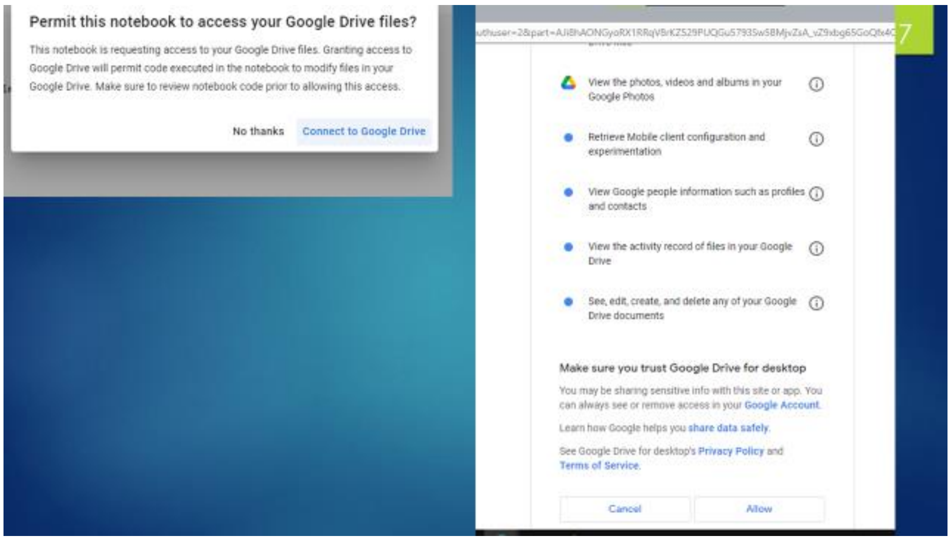
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

$$G = H \star F$$

- ▶ Working with Google Colab



```
#loading All
import numpy as np
import pandas as pd
import cv2 as cv
from google.colab.patches imp
from skimage import io
from skimage import color
from PIL import Image
import matplotlib.pylab as pl
```



Permit this notebook to access your Google Drive files?

This notebook is requesting access to your Google Drive files. Granting access to Google Drive will permit code executed in the notebook to modify files in your Google Drive. Make sure to review notebook code prior to allowing this access.

No thanks | Connect to Google Drive

- View the photos, videos and albums in your Google Photos
- Retrieve Mobile client configuration and experimentation
- View Google people information such as profiles and contacts
- View the activity record of files in your Google Drive
- See, edit, create, and delete any of your Google Drive documents

Make sure you trust Google Drive for desktop

You may be sharing sensitive info with this site or app. You can always see or remove access in your Google Account.

Learn how Google helps you share data safely.

See Google Drive for desktop's Privacy Policy and Terms of Service.

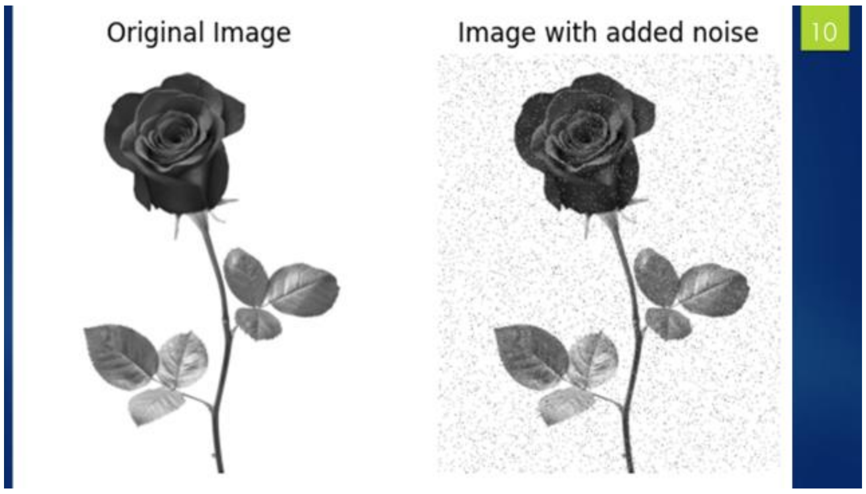
Cancel | Allow

- ▶ **Custom function for adding noise**

I am using a custom function designed for adding salt and pepper noise to an image. You can go to my image filtering repository in GitHub to see and download all codes. Link to my public repository for Image filters is: https://github.com/dratul/ImageFilters.

Code:

```
import random

def add_noise(img):

   row , col = img.shape

# Randomly pick pixels in the image to make it white

 # Pick a random number between 300 and 10000

   number_of_pixels = random.randint(300, 10000)

   for i in range(number_of_pixels):
      # Pick a random y coordinate

      y_coord=random.randint(0, row - 1)
      # Pick a random x coordinate

      x_coord=random.randint(0, col - 1)
      # Color that pixel to white

      img[y_coord][x_coord] = 255
```

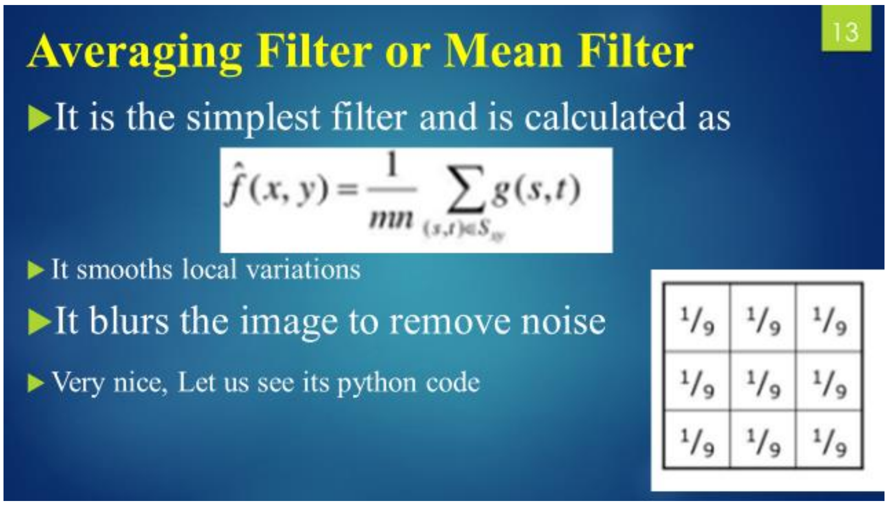


- **Basic Image Filters**
- **Basic filters used in images processing are**
    - Averaging Filter or Mean Filter
    - Median Filter
    - Gaussian Filter
  - Let us start with the first filter, i.e. Mean Filter
- **Averaging Filter**

- It is also known as the smoothing filter. It removes the high-frequency content from the image.
- It is also used to blur an image. Consider the following filtering array, which we'll call a "mean kernel".

- For each pixel, a kernel defines which neighboring pixels to consider when filtering, and how much to weight those pixels.

▶ **Averaging Filter or Mean Filter**



Averaging Filter or Mean Filter

▶ It is the simplest filter and is calculated as

$$\hat{f}(x,y) = \frac{1}{mn} \sum_{(s,t)\in S_{xy}} g(s,t)$$

▶ It smooths local variations

▶ It blurs the image to remove noise

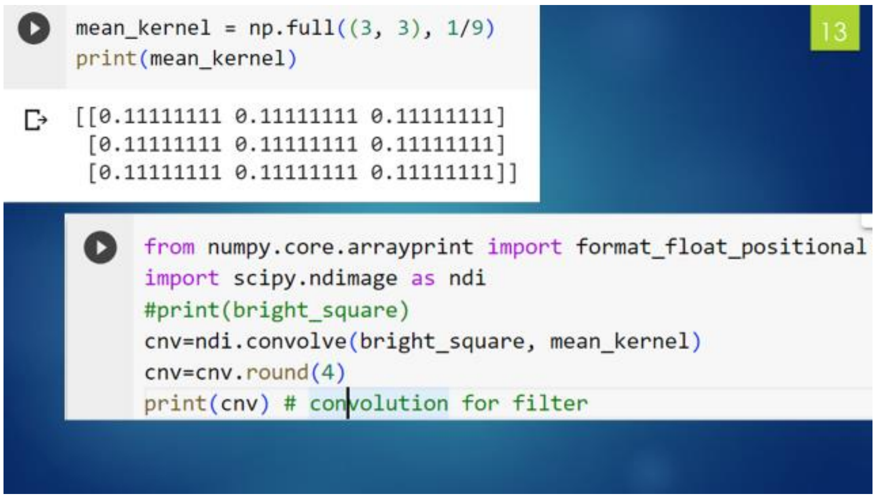
▶ Very nice, Let us see its python code

$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$
$1/9$	$1/9$	$1/9$

▶ It is the simplest filter and is calculated as

▶ It smooths local variations

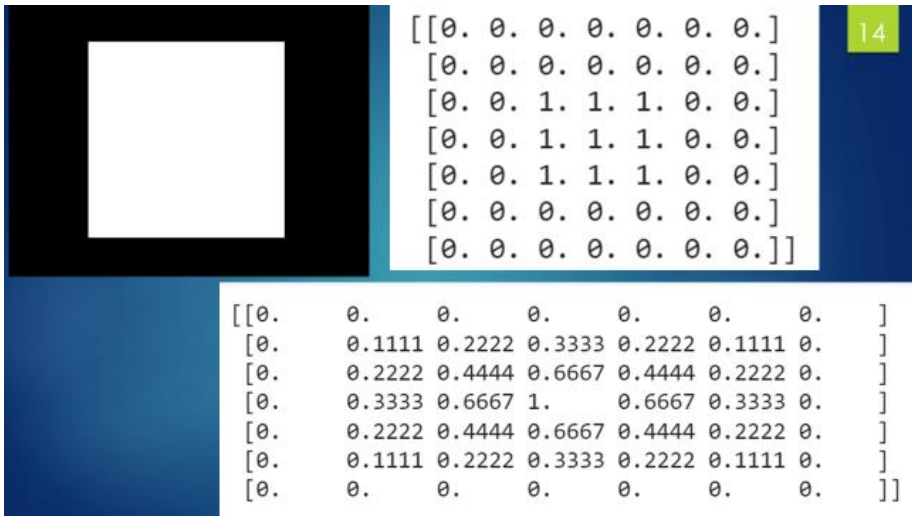
▶ It blurs the image to remove noise



```
mean_kernel = np.full((3, 3), 1/9)
print(mean_kernel)
```

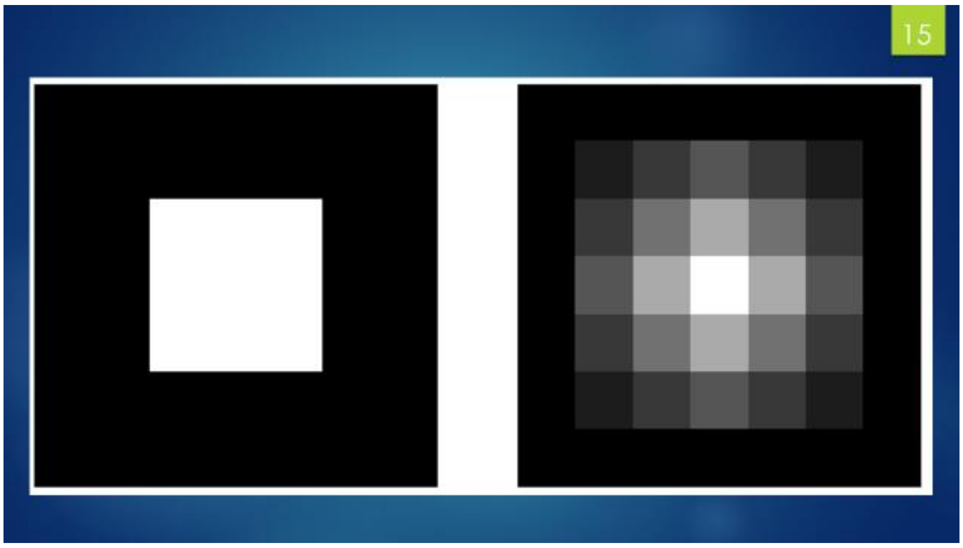
```
[[0.11111111 0.11111111 0.11111111]
 [0.11111111 0.11111111 0.11111111]
 [0.11111111 0.11111111 0.11111111]]
```

```
from numpy.core.arrayprint import format_float_positional
import scipy.ndimage as ndi
#print(bright_square)
cnv=ndi.convolve(bright_square, mean_kernel)
cnv=cnv.round(4)
print(cnv) # convolution for filter
```



```
[[0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 1. 1. 1. 0. 0.]
 [0. 0. 1. 1. 1. 0. 0.]
 [0. 0. 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.    ]
 [0.     0.1111 0.2222 0.3333 0.2222 0.1111 0.    ]
 [0.     0.2222 0.4444 0.6667 0.4444 0.2222 0.    ]
 [0.     0.3333 0.6667 1.     0.6667 0.3333 0.    ]
 [0.     0.2222 0.4444 0.6667 0.4444 0.2222 0.    ]
 [0.     0.1111 0.2222 0.3333 0.2222 0.1111 0.    ]
 [0.     0.     0.     0.     0.     0.     0.    ]]
```

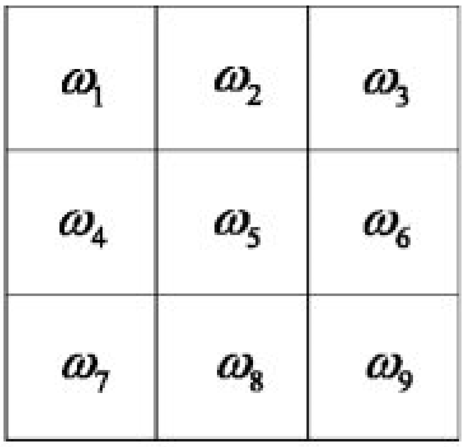


Very nice, Let us see its python code

```
# Convolve the 3X3 mask over the image

img_new = np.zeros([m, n])

for i in range(1, m-1):

    for j in range(1, n-1):

            temp = img[i-1, j-1]*mask[0, 0]+img[i-1, j]*mask[0, 1]+img[i-1, j + 1]*mask[0, 2]+img[i, j-1]*mask[1, 0]+ img[i, j]*mask[1, 1]+img[i, j + 1]*mask[1, 2]+img[i + 1, j-1]*mask[2, 0]+img[i + 1, j]*mask[2, 1]+img[i + 1, j + 1]*mask[2, 2]

    img_new[i, j]= temp

    img_new = img_new.astype(np.uint8)
```

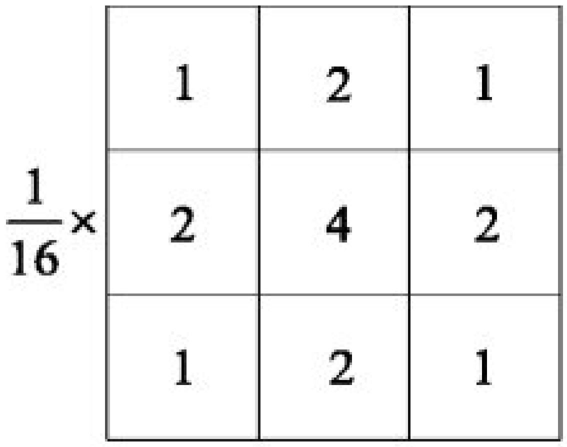
▶ **Weighted Averaging Filter**

Instead of using ‘1’ at each point of the kernel a weighted kernel is used



ω_1	ω_2	ω_3
ω_4	ω_5	ω_6
ω_7	ω_8	ω_9

(a) A representation of a general 3×3 filter mask



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

1	2	1
2	4	2
1	2	1

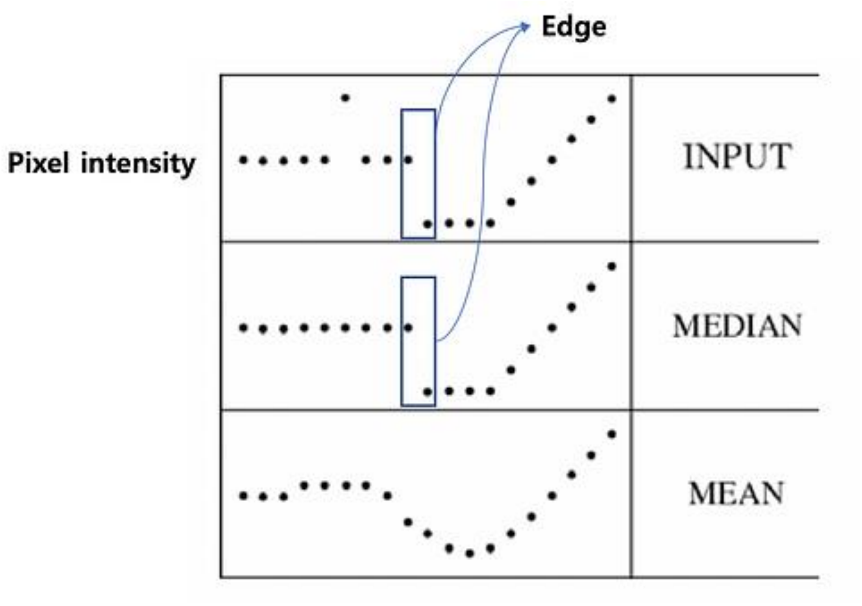
(b) A 3×3 weighted average filter mask

▶ **Median Filters**

- Median Filter is also used for smoothing.
  Its basic idea is to replace each pixel by the median of its neighboring pixels.
- This filter wonderfully removes spikes introduced by noise, especially impulse and salt & pepper noise.
- Another advantage of median filter is that it does not introduce new pixel values since it only re-use existing pixel values from window.

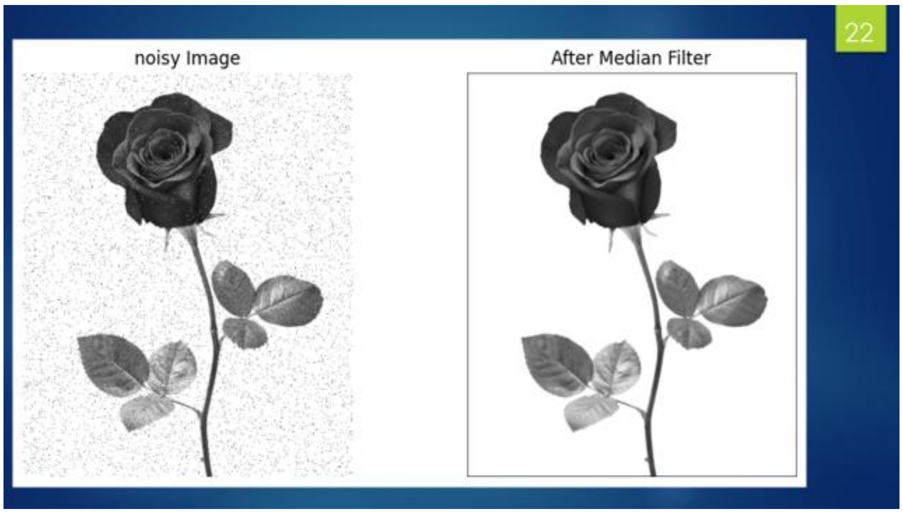
▶ **Median Filters**

▶ Further, unlike other averaging filters, it remove noise without losing edge information as is illustrated below.



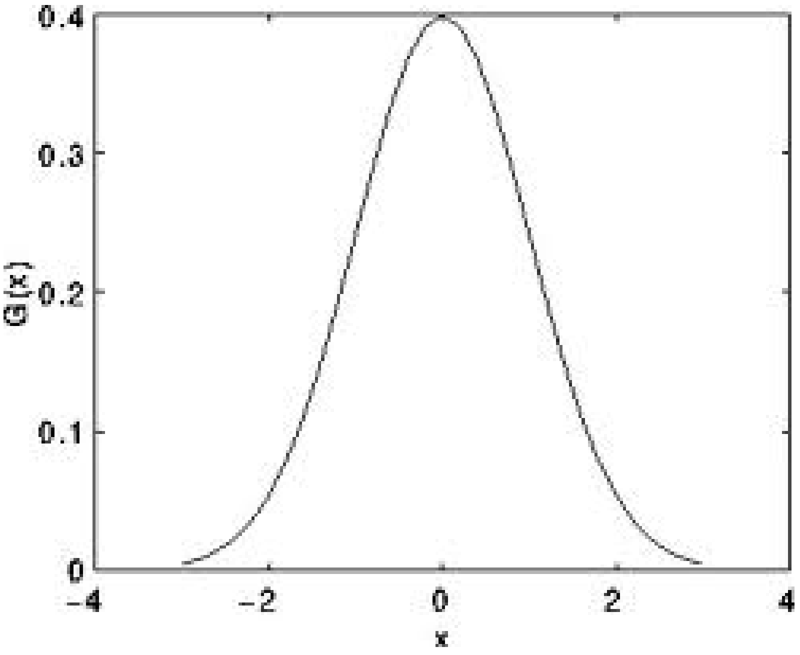
▶ # Let us write our own Median Spatial Domain Filtering

- # Read the image
- img_noisy1 = img_O
- # Obtain the number of rows and columns # of the image
- m, n = img_noisy1.shape
- # Traverse the image. For every 3X3 area,
- # find the median of the pixels and
- # replace the center pixel by the median
- img_new1 = np.zeros([m, n])
- Median Filters using OpenCV
- Open CV Library has built in function for median filtering therefore we can directly use it as follows
- #Median Filter by using in built function of open CV
- img_O = img_n #cv.imread('/content/drive/MyDrive/Colab Notebooks/rose.jpg')
- img = cv.cvtColor(img_O, cv.COLOR_BGR2RGB);
- img = color.rgb2gray(img);
- img_median = cv.medianBlur(img_O, 5) # Add median filter to image
- img_m = cv.cvtColor(img_median, cv.COLOR_BGR2RGB);
- img_m = color.rgb2gray(img_m);
- imshow_all(img, img_m,titles=['original', 'median filtered']) # Display img with median filter



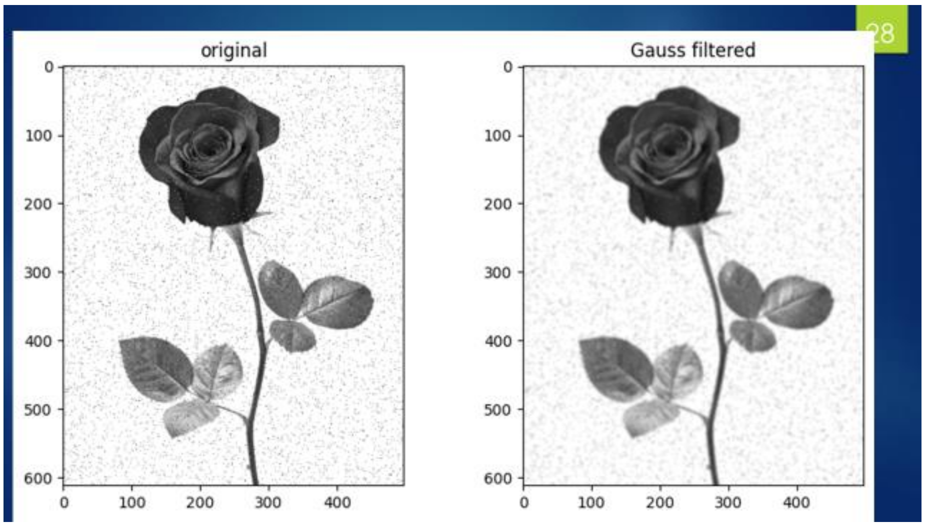
- **Gaussian Filter**
- Now we are at the discussion of most useful filter (although not the fastest).
- Gaussian filtering is done by convolving each point in the input array with a Gaussian kernel and then summing them all to produce the output array.
- Gaussian Filter
- Just to make the picture clearer, do you remember how a 1D Gaussian kernel look like?
- Very nice, now if we extend that equation to 2D filters then it can be represented as

$$G_0(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x-\mu_x)^2+(y-\mu_y)^2}{2\sigma_x^2\,\sigma_y^2}}$$



- **#Gausian Filter by using in built function of open CV**
  - img_O = img_n #cv.imread('/content/drive/MyDrive/Colab Notebooks/rose.jpg')

- img = cv.cvtColor(img_O, cv.COLOR_BGR2RGB);
- img = color.rgb2gray(img);
- img_gauss = cv.GaussianBlur(img_O, (5,5),cv.BORDER_DEFAULT)
  # Add Gauss filter to image
- img_m = cv.cvtColor(img_gauss, cv.COLOR_BGR2RGB);
- img_m = color.rgb2gray(img_m);
- imshow_all(img, img_m,titles=['original', 'Gauss filtered'])



▶ **Conclusion**

▶ how to use Google Colab,

▶ what is noise, how to introduce a calculated noise to images, what are image filters, process of filtering, application of filters in image denoising and some basic filters like mean or averaging filter, median filter and gaussian filter.

▶ All codes will be available in my Github repository of image filters

▶ Thank You all .