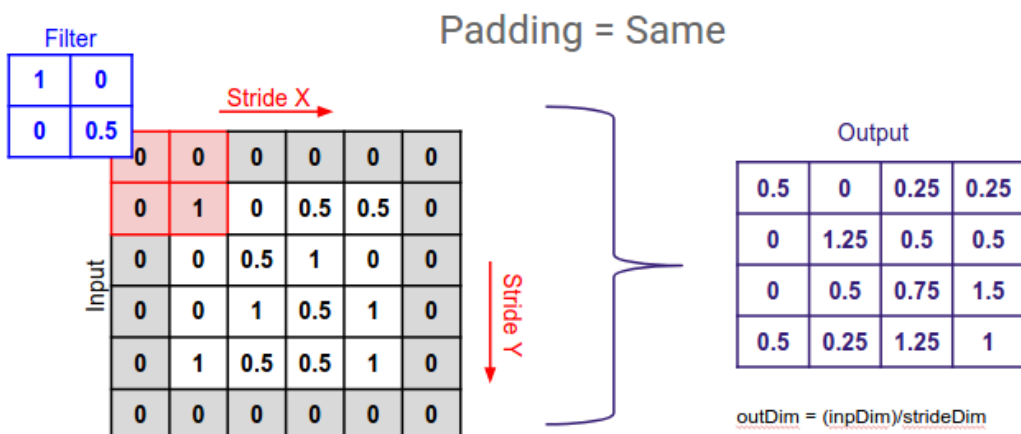


Q1) Explicitly describe image convolution:
the input, the transformation, and the output.
Why is it useful for computer vision?

To perform convolution, we need to have a kernel, which is a filter in our case, that would be convolved with the image.

Convolution is basically an integration. A filter/kernel is placed on an image and all the elements of the kernel are multiplied to the corresponding elements of the image resulting with a number which is placed at the output at a location corresponding to the center of the kernel.

If we need to obtain an output of the same size as the image, we would need to pass the image in a way that would center the filter with a cell corresponding to the first cell of the image. This could be shown by the image below. Zero padding around the image, so when the filter is placed on it and multiplication is done with the corresponding elements the result could be placed in the center, which in case of an even kernel could be assumed to be the bottom right cell for the image below. After performing the calculation for the first part, the filter then moves with a step called stride. Stride means by how many cells should the filter move, for the following project, we used stride=1 in both the x, and y directions.



The process of placing a filter, multiplying the kernel with the corresponding part of the image, summing the results into a single number, and placing this number at the output then moving the kernel with a stride could be represented by the following mathematical equation.

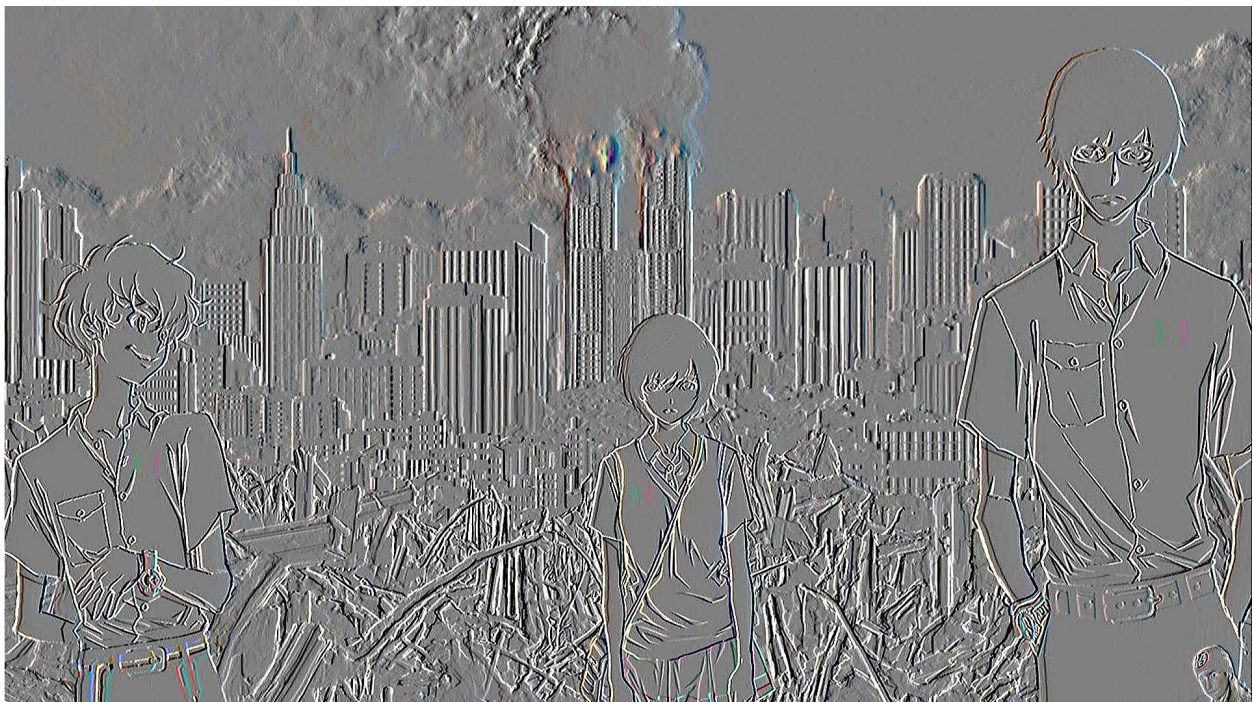
$$(f * g)(t) := \int_{-\infty}^{\infty} f(\tau)g(t - \tau) d\tau.$$

So how could this process be useful when it comes to computer vision? Dot product between the filter and a small region of the input image at each pixel location results in a value placed in the output matrix through the process described above. In computer vision this output operation when done on images enhances or extracts features from the image.

Sobel filter can extract edges



When the image above is passed through a sobel filter, we would obtain the image: $\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$

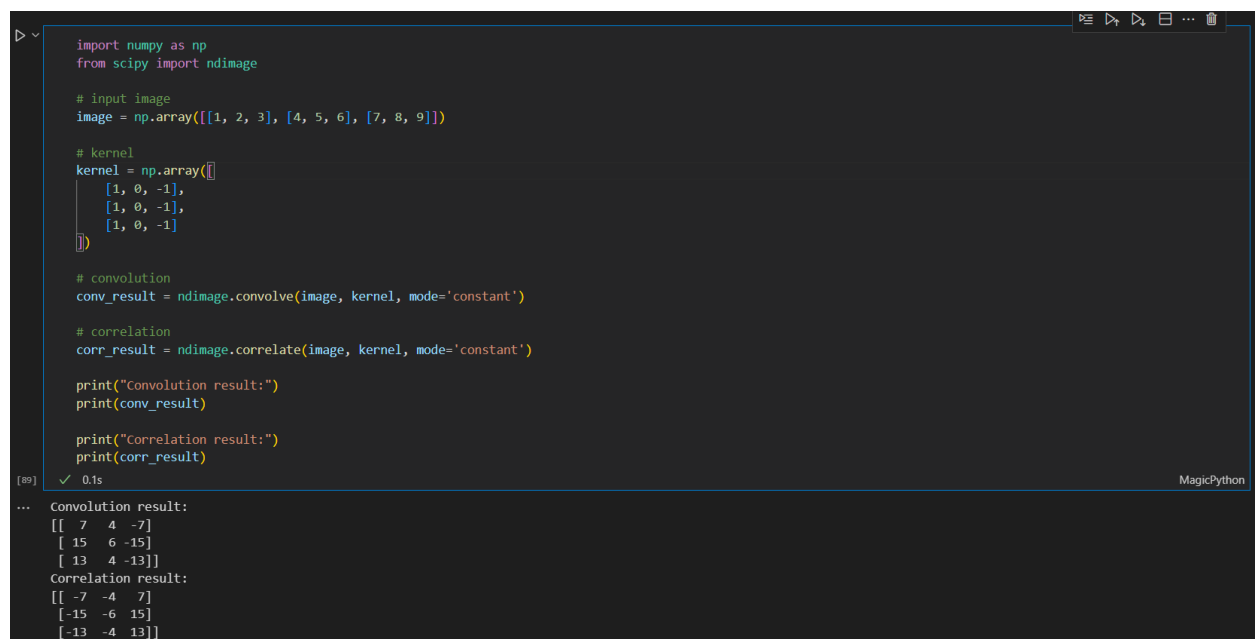


It is commonly used in image processing tasks such as image filtering, edge detection, image enhancement, and feature extraction. Convolutional Neural Networks (CNNs) also heavily rely on convolution to learn meaningful representations of images for tasks such as object recognition, segmentation, and classification.

Q2) What is the difference between convolution and correlation?
Construct a scenario which produces a different output between both operations.

The difference between convolution and correlation is that correlation is 180 deg rotated from convolution.

Scenario: When the kernel is not symmetric or when the input signal has some inherent symmetry that impacts the operation's result, convolution and correlation can provide different values.



```
import numpy as np
from scipy import ndimage

# input image
image = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# kernel
kernel = np.array([
    [1, 0, -1],
    [1, 0, -1],
    [1, 0, -1]
])

# convolution
conv_result = ndimage.convolve(image, kernel, mode='constant')

# correlation
corr_result = ndimage.correlate(image, kernel, mode='constant')

print("Convolution result:")
print(conv_result)

print("Correlation result:")
print(corr_result)
```

[89] ✓ 0.1s

... Convolution result:
[[7 4 -7]
 [15 6 -15]
 [13 4 -13]]

Correlation result:
[[-7 -4 7]
 [-15 -6 15]
 [-13 -4 13]]

Q3) What is the difference between a high pass filter and a low pass filter in how they are constructed, and what they do to the image? Please provide example kernels and output images

A low pass filter attenuates high-frequency frequencies while allowing low frequency signals to pass through. In terms of image processing, this means that the filter eliminates noise or high-frequency features while maintaining the smoothness of the image.

On the other hand, a high pass filter attenuates low-frequency frequencies while allowing high-frequency signals to flow through. By reducing the smoothness or low-frequency components, the filter in image processing enhances the edges and details in the image.

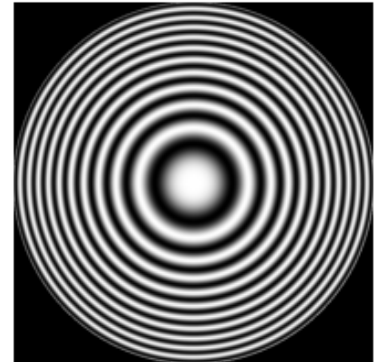
Low Pass Filter:



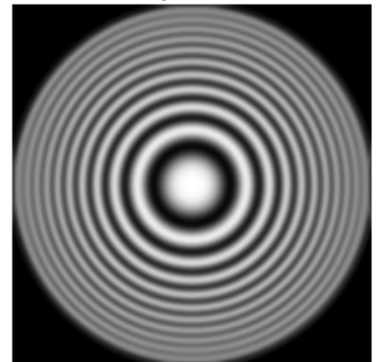
High Pass filter = Image- Low pass filter



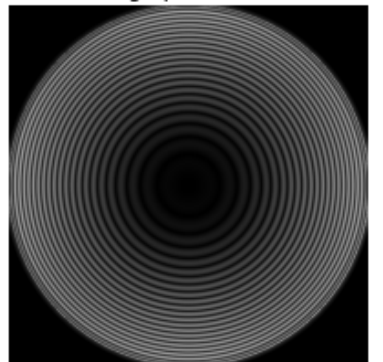
Original Zone plate



Low pass filter



High pass filter



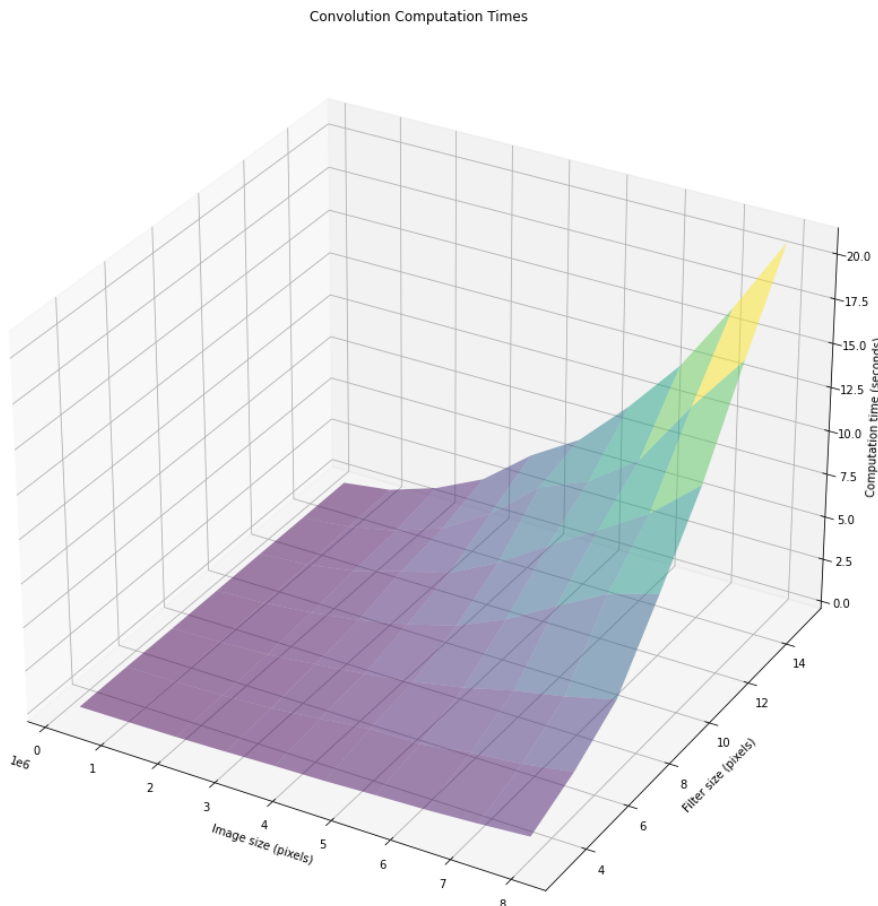
Low pass filter: High pass filter

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad w = 1/9 \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

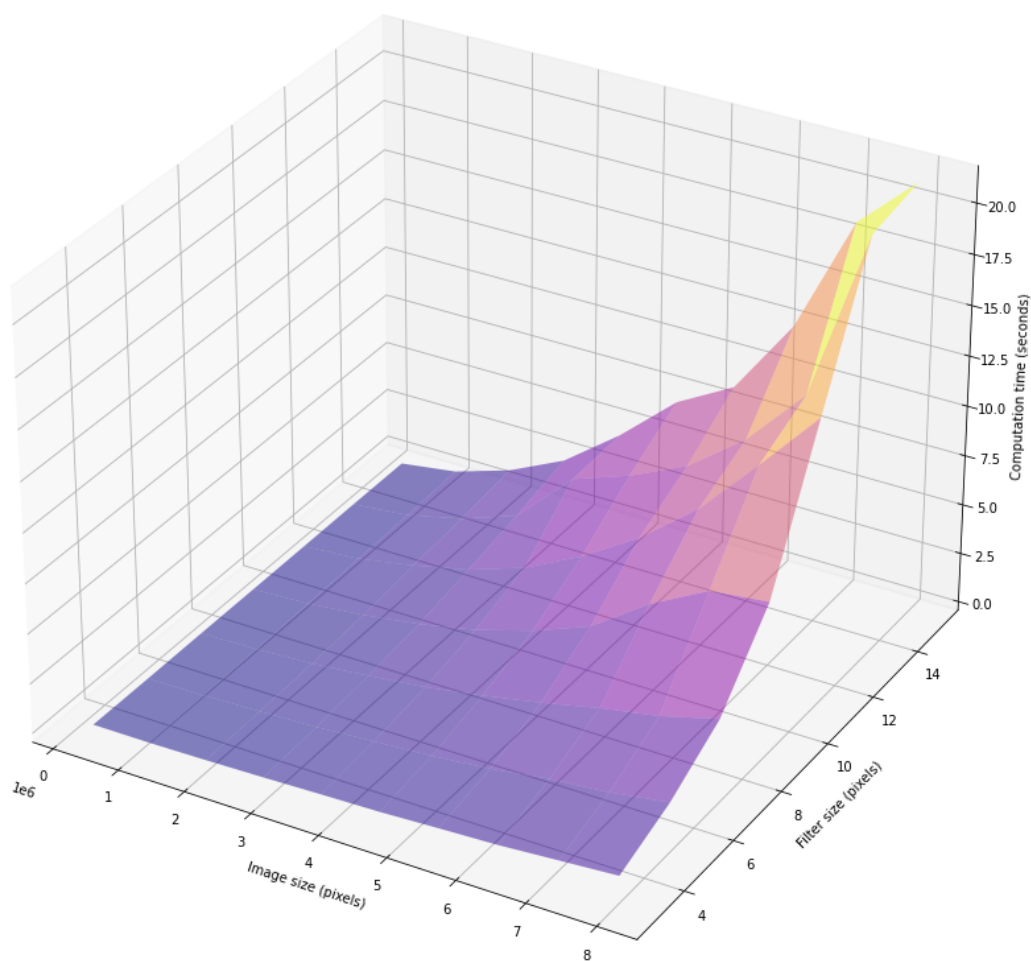
Q4)How does computation time vary with filter sizes from 3×3 to 15×15 (for all odd and square sizes), and with image sizes from 0.25 MPix to 8 MPix (choose your own intervals)? Measure both using `scipy.ndimage.convolve` or `scipy.ndimage.correlate` to produce a matrix of values. Use the `skimage.transform` module to vary the size of an image. Use an appropriate charting function to plot your matrix of results, such as `Axes3D.scatter` or `Axes3D.plot_surface`. Do the results match your expectation given the number of multiply and add operations in convolution? **The code is attached with the project.(Q4.py)**

As the filter size increases, the computation time increases as well. In addition, as the image size increase, the computation time increases as well.

Below is a demonstration of the results:



Correlation Computation Times



Yes, as the sizes of filters and pixels of the image increase, the number of multiplications and additions increase, resulting in larger computation times.