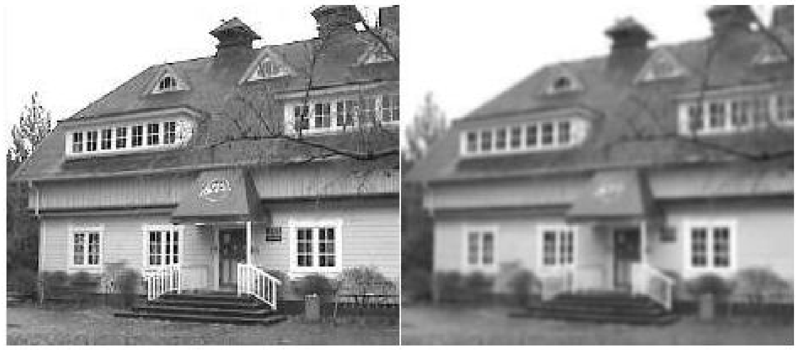
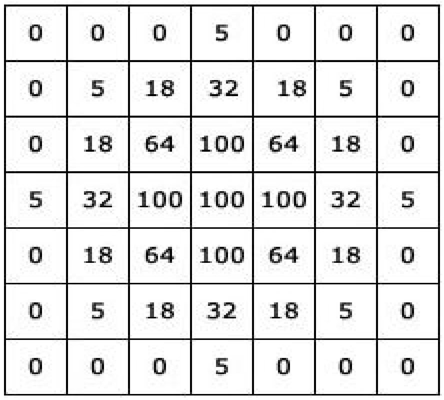
Q1: Explicitly describe image convolution: the input, the transformation, and the output. Why is it useful for computer vision?
A1:

- The input to convolution is two signals. So, in image convolution, the first signal is an image and the second is a filter (or kernel).
- The transformation:
  To apply the kernel on the image, the kernel (after being flipped) is slided over each pixel in the image. Then, the overlapping entries from the image and kernel are multiplied and added. The result is then placed in a pixel in the output image that corresponds to the center of the filter.
- The output is a new modified filtered image.

Image convolution is useful for computer vision as with different combinations of the filter's size and weight, different effects can be noticed on the image. Furthermore, we can enhance, modify or extract useful information from the photos such as edges, corners or blobs.
Examples of applying different filters: source:

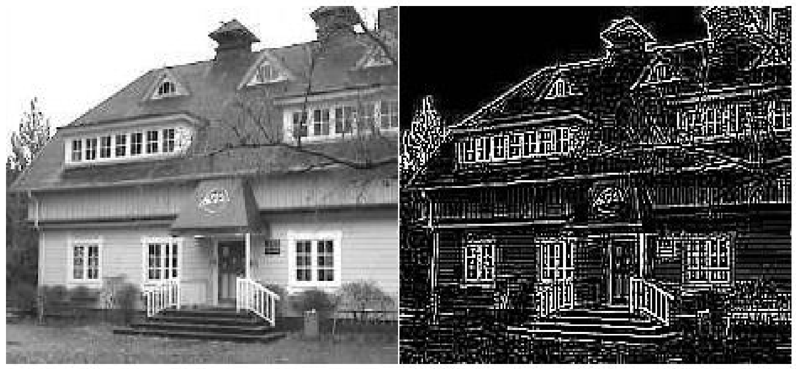
- Blurring with a gaussian filter:

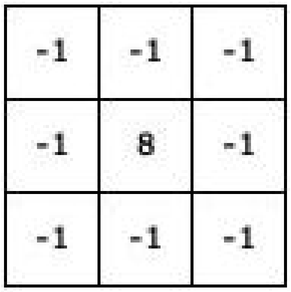




0	0	0	5	0	0	0
0	5	18	32	18	5	0
0	18	64	100	64	18	0
5	32	100	100	100	32	5
0	18	64	100	64	18	0
0	5	18	32	18	5	0
0	0	0	5	0	0	0

- Edge Detection:





-1	-1	-1
-1	8	-1
-1	-1	-1

Source: https://aishack.in/tutorials/image-convolution-examples/

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

A2: Both operations are shift-invariant and linear. Moreover, their effect is the same for every pixel in the image; and each pixel is replaced with a linear combination of itself and its neighbours.

- Correlation is a measurement of the similarity between two signals. So, it measures how a portion of the image looks like the filter.

$$F \circ I(x) = \sum_{i=-N}^{N} F(i) I(x+i)$$

- Convolution is a measurement of the effect one signal (as the kernel) has on another (as an image).
  The mathematical formula of convolution is the same as that of the correlation, except that we should flip the filter by 180 degrees before applying it on the image. So, the output of convolution is a flipped version of the output of correlation. However, if the filter is symmetric, the two operations will have identical results.

$$F * I(x) = \sum_{i=-N}^{N} F(i) I(x-i)$$

  In case of 2-D convolution, the filter is flipped both horizontally and vertically.

$$F * I(x,y) = \sum_{j=-N}^{N} \sum_{i=-N}^{N} F(i,j) I(x-i, y-j)$$

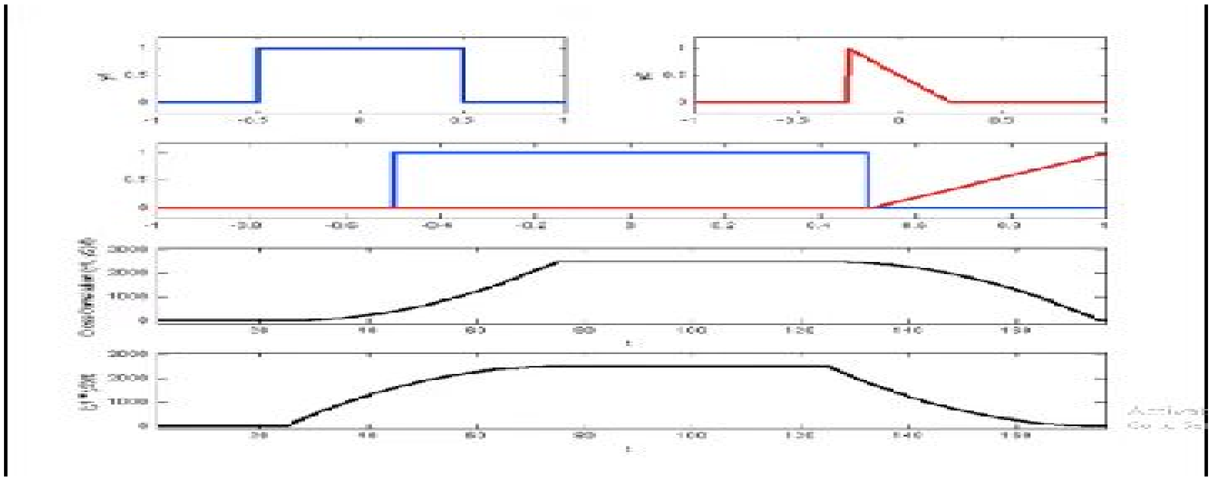
  Another difference is that convolution is an associative operation and correlation is not. Hence, we can combine different filters together and obtain the desired super-filter. However, in correlation, the order with which the filters are applied changes the effect.

In general, correlation is useful when we are performing image-matching, or pattern recognition; and convolution is useful when applying different filters on an image.

An example when the filter is not symmetric:

The first is the result of correlation, and the second is the result of convolution.

source: https://www.youtube.com/watch?v=Ma0YONjMZLI

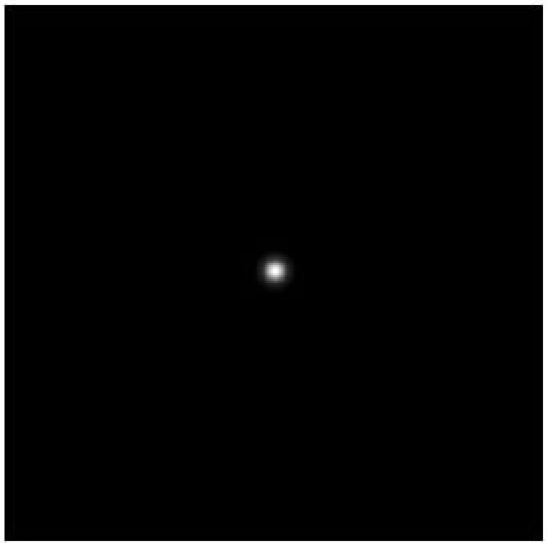


Q3: What is the difference between a high pass filter and a low pass filter in how they are constructed, and what they do to the image? Please provide example kernels and output images.

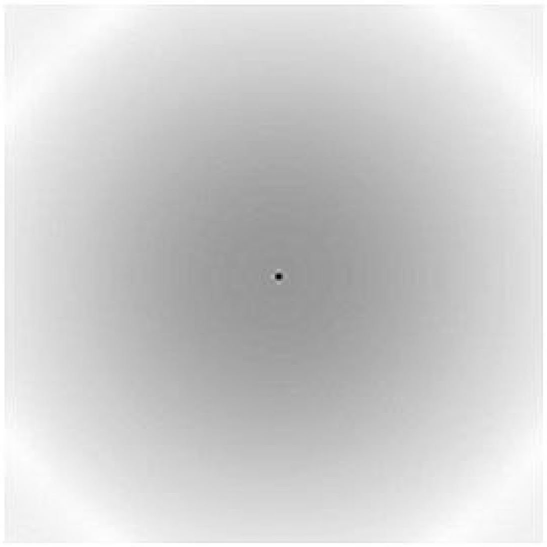
A3:

- The low-pass filter attenuates the high-frequency components in an image while preserving the low-frequency components. It is applied when we want to smooth the image. Low-pass filters can also help in denoising an image as the noise is a high-frequency component.
  The high-pass filter attenuates the low-frequency components in an image while preserving the high-frequency components. It is applied when we want to sharpen the image or perform edge detection.
- The low-pass filter attenuates frequencies that are further away from the center; while the high-pass filter attenuates frequencies that are near to the image center. Hence, in construction, the high-pass filter has the least values at the center; while the low-pass filter has the least values away from the center.

- Low-pass in spatial domain:

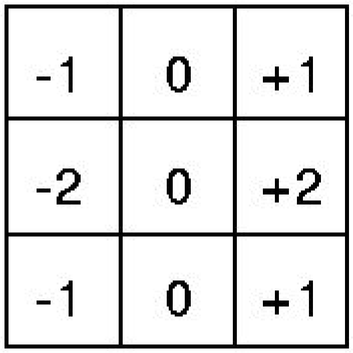


- High-pass in spatial domain:



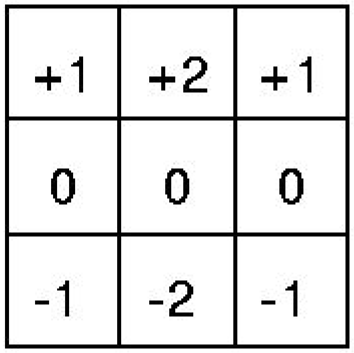
- ***Examples of High-pass filters:***
    1. Sobel high-pass filter:
       This filter is used for edge detection; Gx is designed to respond maximally to edges running vertically; while Gy is designed to respond maximally to edges running horizontally. They can be combined together to find the absolute magnitude at each point in the image.

$$|G| = \sqrt{Gx^2 + Gy^2}$$



-1	0	+1
-2	0	+2
-1	0	+1

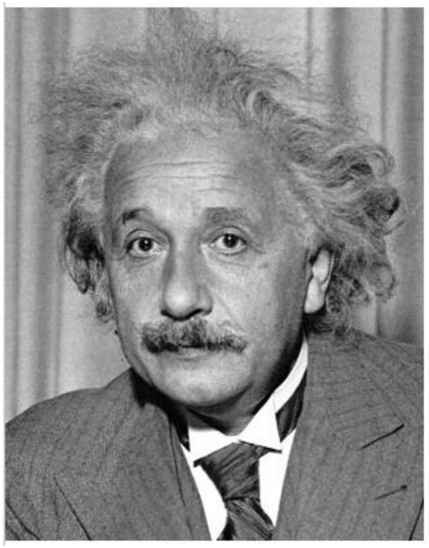
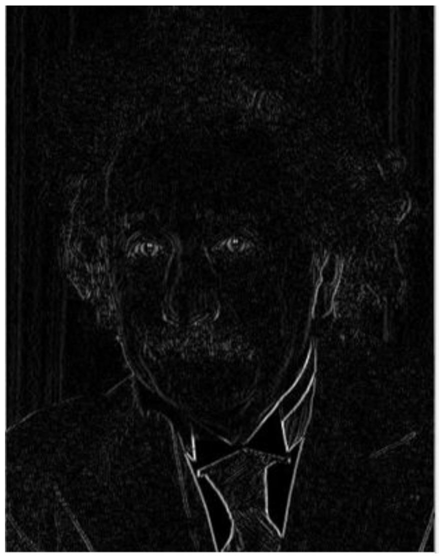
Gx

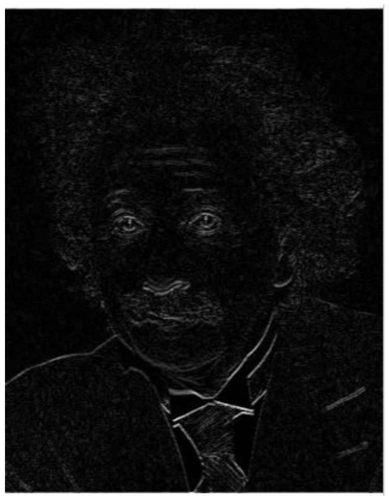


+1	+2	+1
0	0	0
-1	-2	-1

Gy

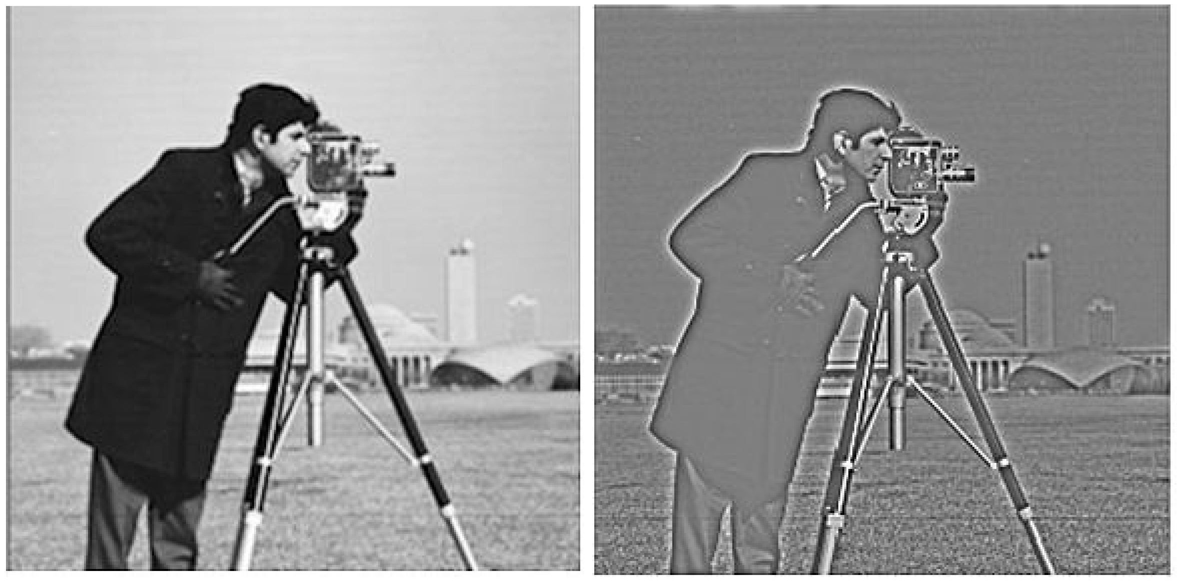
Source: https://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm



->Applying Gx


-> Applying Gy

2. Gaussian High-pass filter:
   It attenuates components near the image center: (W/2 and H/2)

$$X'_{uv} = X_{uv}\left(1 - e^{-\frac{\left(\frac{H}{2}-u\right)^2 + \left(\frac{W}{2}-v\right)^2}{A^2}}\right)$$

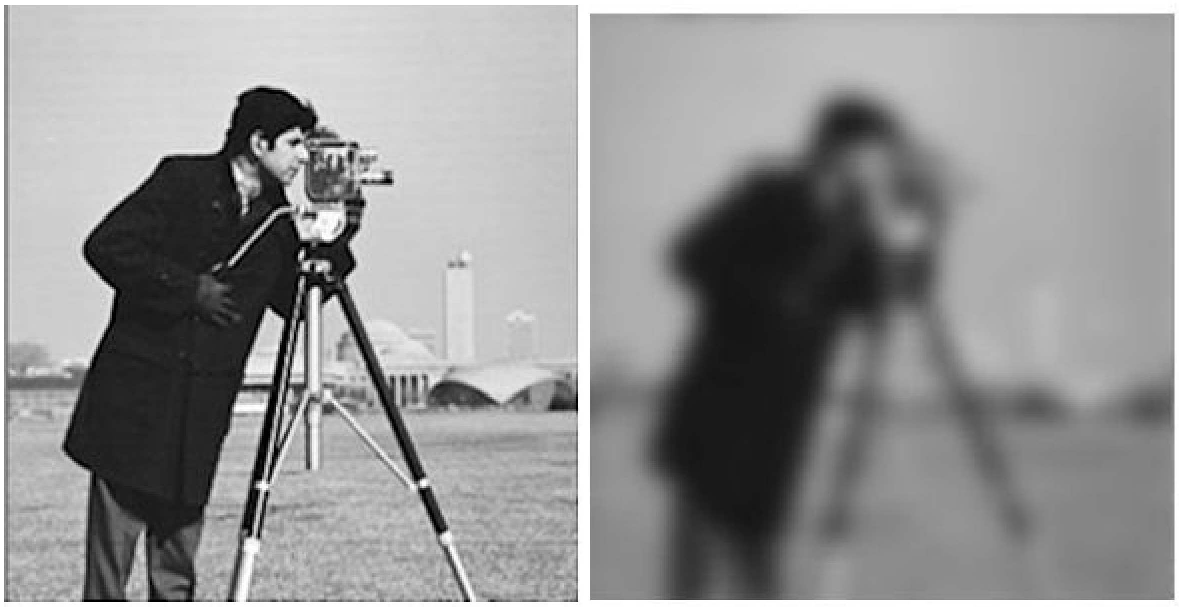


Source:
https://biomedpharmajournal.org/vol7no2/image-sharpening-by-gaussian-and-butterworth-high-pass-filter/

In the gaussian high-pass filter, it does the same job as an ideal high-pass filter. However, the transition is smoother compared to the ideal one.

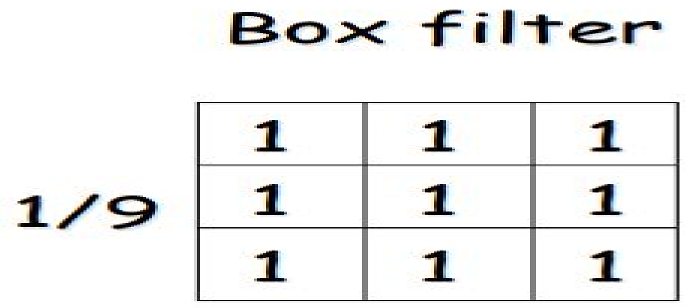
- ***Examples of Low-pass filters:***
    1. Gaussian Low-pass filter:
       It attenuates frequency components further away from the center (H/2 and W/2).

$$X'_{uv} = X_{uv} e^{-\frac{\left(\frac{H}{2}-u\right)^2+\left(\frac{W}{2}-v\right)^2}{A^2}}$$



Source:
https://biomedpharmajournal.org/vol7no2/image-sharpening-by-gaussian-and-butterworth-high-pass-filter/

2. Box (Mean) Filter:
   It replaces each pixel with an average of its neighbourhood. This blurs the image and decreases the noise within it.



Box filter

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

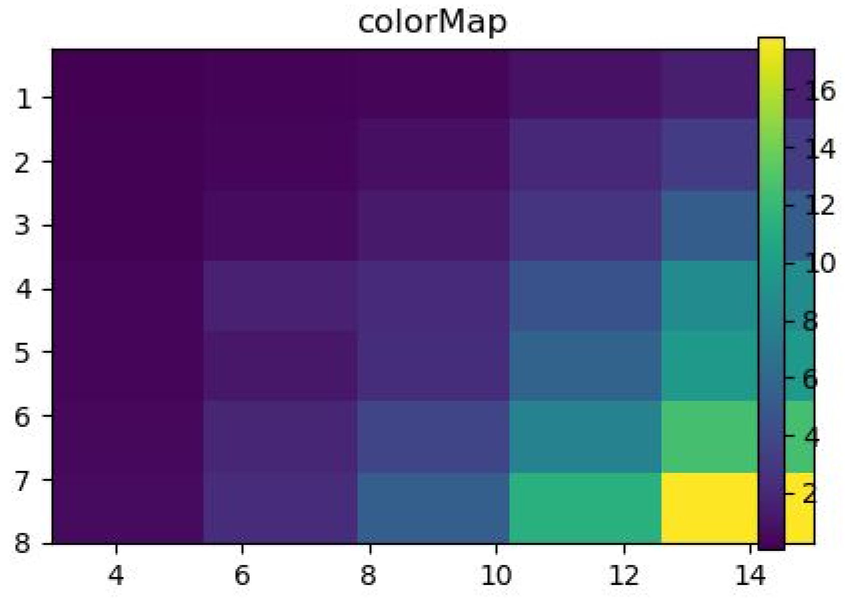
- Convolving an image with a 11x11 box filter

Source: http://www.cse.psu.edu/~rtc12/CSE486/lecture04.pdf

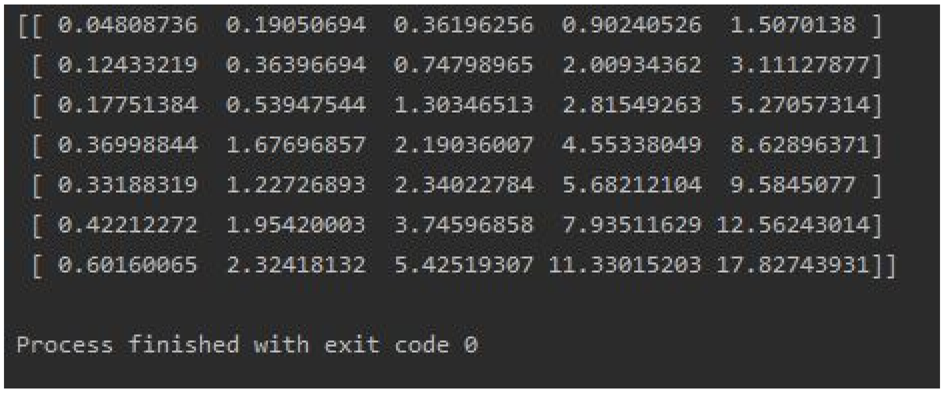
- The problem with box-filter is that increasing the filter size results in losing the image's detail. This is why the low-pass gaussian filter is preferred as we do not lose as much detail as in the box-filter.

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.scatterorAxes3D.plot surface. Do the results match your expectation given the number of multiply and add operations in convolution?
Image: RISDance.jpg (in the .tex directory).

A4: a color map of the time taken to perform a gaussian filter on different size images is shown below:
On the X axis is the kernel sizes, on the y axis is the image size in Megapixel.
We see, as expected, that the running time increases with increasing both kernel and image size:



The time matrix values:



```
[[ 0.04808736  0.19050694  0.36196256  0.90240526  1.5070138 ]
 [ 0.12433219  0.36396694  0.74798965  2.00934362  3.11127877]
 [ 0.17751384  0.53947544  1.30346513  2.81549263  5.27057314]
 [ 0.36998844  1.67696857  2.19036007  4.55338049  8.62896371]
 [ 0.33188319  1.22726893  2.34022784  5.68212104  9.5845077 ]
 [ 0.42212272  1.95420003  3.74596858  7.93511629 12.56243014]
 [ 0.60160065  2.32418132  5.42519307 11.33015203 17.82743931]]

Process finished with exit code 0
```