Mini-Project 1 Questions
CIE 552 | Spring 2021

Instructions

- 4 questions.
- Write code where appropriate.
- Feel free to include images or equations.

Questions

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

A1:

Images can be thought of as signals and signals can be decomposed into a sum of scaled and shifted impulse functions. Convolution is a mathematical operation on two functions that produces a function that expresses how the shape of one is modified by the other.

The output of this linear and time-invariant system can be described by the convolution of an input signal $x[m,n]$ and an impulse response *(the kernel or the mask)* $h[m,n]$.

$$y[m,n] = x[m,n] * h[m,n] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} x[i,j] \cdot h[m-i,\ n-j]$$

Normally, the kernel is of the order 1x1, 3x3, 5x5, 7x7. And, it should be of an odd number, because we won't be able to find the mid of the kernel otherwise. As for the output, it is noted that it is a sum of elements of scaled and shifted impulse responses.

How to perform it on an image?

- ➔ Flipping the kernel vertically and horizontally.
- ➔ Padding the image to counter the border effect and have the output be same size as input.
- ➔ Sliding the kernel onto the input image.
- ➔ Multiplying the corresponding elements and summing them.
- ➔ Repeating the process till all values of the output image are calculated.

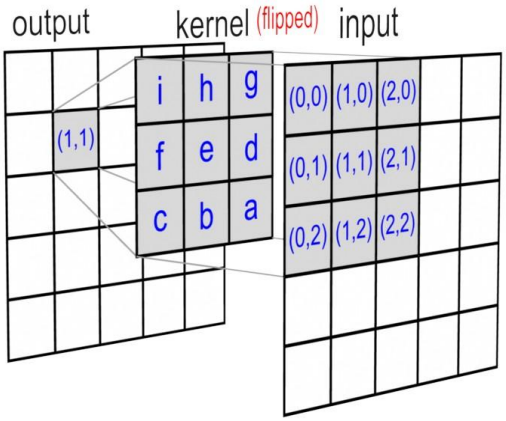

Fig.1. 2D Image Convolution [1]

Why is it useful in computer vision?

It is essentially a filter that could be applied to images for all sorts of purposes. Convolving a kernel [convolution matrix] with an image can be used for sharpening the image, blurring it, edge detection and a lot more.

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

Please use scipy.ndimage.convolve and scipy.ndimage.correlate to experiment!

A2:

Correlation mainly measures how similar two signals are while ***convolution*** measures the effect one signal has on the other.

As for the mathematical calculation of the two, they are practically the same in the time domain. Convolution is correlation with the kernel flipped vertically and horizontally which means they are exactly the same when the kernel is symmetric. However, when the kernel is not symmetric, they are very different operations. Convolution translates to multiplication in the frequency domain; making it an associative operation. This allows us to apply multiple kernels successively and get an image with all their effects. Correlation does not have such property since it is just multiplication by the complex conjugate in the frequency domain. Yet, it is useful in some cases; such as template matching.

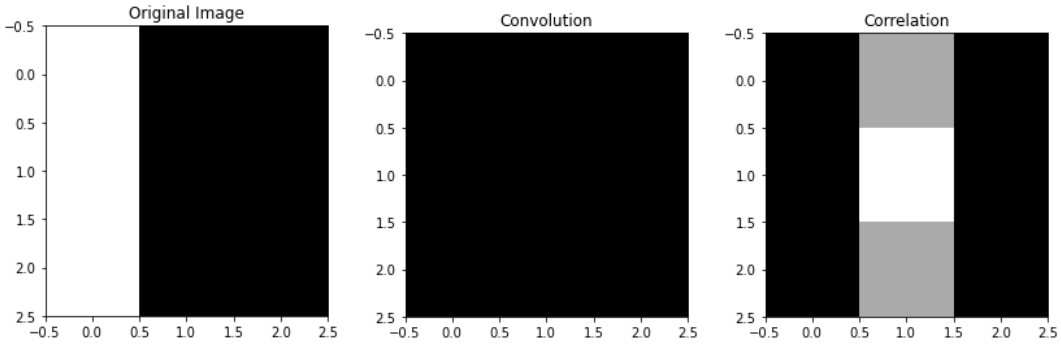

Fig.2. Comparison between convolution and correlation.

As an example, we take the image above which is a simple 3x3 kernel that is,

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Then, we correlate and convolve it with itself. The convolution and correlation outputs vary. In the correlated image, there is a point of maxima representing perfect matching which makes sense considering the image is correlated with itself. The same technique used in template matching.

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:

A low pass filter is one that attenuates high frequencies while passing low frequencies. They are generally used for ***smoothing and noise removal***. Whereas, a high pass filter is one that does not affect high frequencies. This type of filters is generally used for ***sharpening and removal of aliasing effect***. The following example is an image convolved with a gaussian kernel.

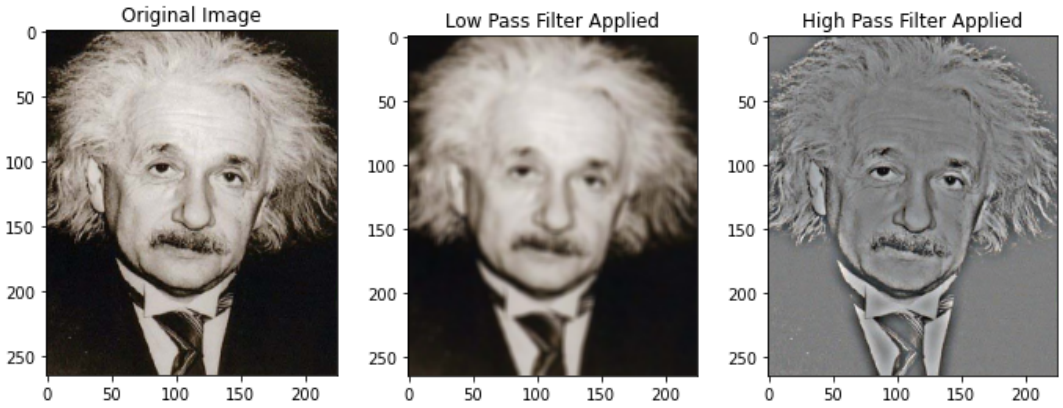

Fig.3. Comparing high and low pass filtered image [gauss].

Now, trying with a typical low pass and a high pass filters,

$$\begin{pmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{pmatrix} \qquad \begin{pmatrix} -1/9 & -1/9 & -1/9 \\ -1/9 & 8/9 & -1/9 \\ -1/9 & -1/9 & -1/9 \end{pmatrix}$$

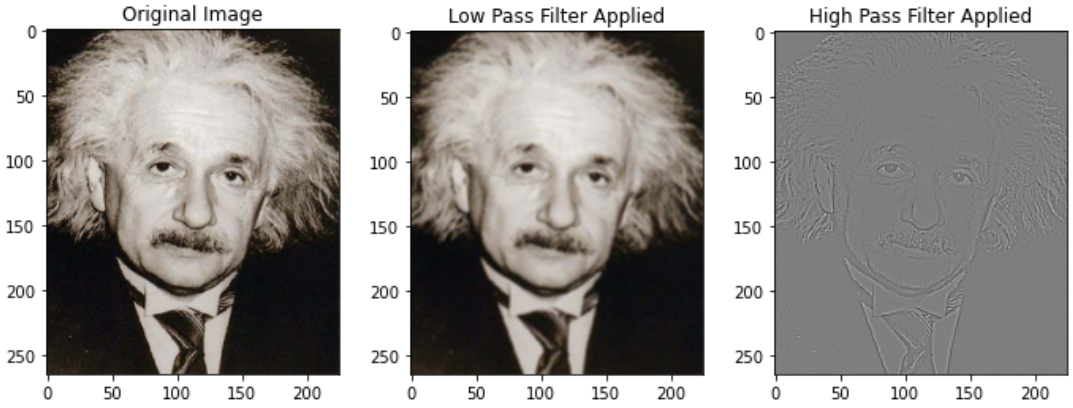

Fig.4. Comparing high and low pass filtered image.

Q4: How does computation time vary with filter sizes from 3x3 to 15x15 (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 *Axes3D.scatter* or *Axes3D.plot_surface*.

Do the results match your expectation given the number of multiply and add operations in convolution?

Image: RISDance.jpg (in the project directory).

A4:

Convolution is a computationally expensive operation. So, it's expected to get slower and slower with larger kernels and images. Our results align with this assumption [2],

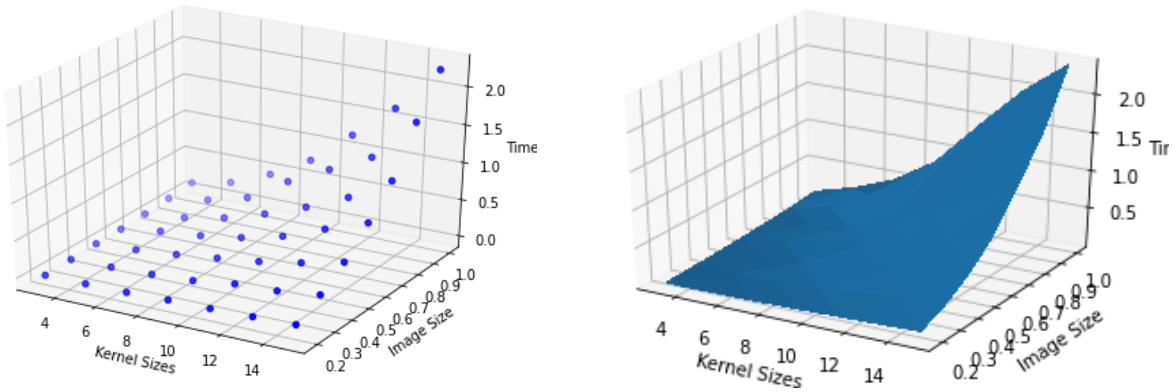

Fig.5. Comparison between convolution and correlation.

The function is taking a kind of an exponential rise as the image gets larger and the kernel gets bigger which is expected. Because convolution is such a crucial operation and we need it to be done at a computationally reasonable time. That's why there are many methods done to speed it up; such as FFT, DFT, factoring kernels, or checking kernel's separability.

References

[1] S.Ahn, "Convolution", Songho.ca, 2005. [Online]. Available: http://www.songho.ca/dsp/convolution/convolution.html. [Accessed: 10- Apr- 2021].

[2] Code we have written for this part:

```
sigma = 7
test_image = load_image('../writeup/RISDance.jpg')
test_image = test_image[:,:,0]
print(test_image.shape)
ksizes = np.arange(3, 16, 2)
ratios = np.linspace(0.185, 1, num=7, endpoint=True)

x_ksize, y_ratio, times = [], [], []
for ratio in ratios:
    for ksize in ksizes:

        image = rescale(test_image, ratio, mode='reflect')
        kernel = create_gaussian_filter(ksize, sigma)
        x_ksize.append(ksize)
        y_ratio.append(ratio)
        start = timer()
        convolve(image, kernel, mode='constant')
        end = timer()
        times.append(end - start)

fig = plt.figure()
ax = Axes3D(fig)

# Plot the values
ax.scatter(x_ksize, y_ratio, times, c = 'b', marker='o')
ax.set_xlabel('Kernel Sizes')
ax.set_ylabel('Image Size')
ax.set_zlabel('Time')
plt.show()
```