The convolution theorem

We have seen that we can filter images by either a convolution operation or by weighting the frequency components of an image in the frequency domain. So, it might seem that there ought to be a relation between convolution and frequency-domain filtering.

In fact, there is an intimate relation between the two, which we call the convolution theorem. Briefly stated, the convolution theorem tells us that the Fourier transform of a convolution is the product of the transforms of the two convolved functions. Symbolically,

$$f * g \supset F \cdot G$$

Proof. Begin with the convolution integral

$$f(x) * g(x) = \int_{-\infty}^{\infty} f(x')\, g(x-x')\, dx'$$

Let $\mathcal{F}(\,)$ be the Fourier transform operation, then

$$\mathcal{F}\left(f(x) * g(x)\right) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x')\, g(x-x')\, dx'\; e^{-j2\pi xs}\, dx$$

Interchange the order of integration and group the x-terms together:

$$= \int_{-\infty}^{\infty} \underbrace{\int_{-\infty}^{\infty} g(x-x')\, e^{-j2\pi xs}\, dx}_{\text{Note this is } \mathcal{F}(g(x-x'))} f(x')\, dx'$$

We note that the indicated term in the integral is the Fourier transform of $g(x-x')$, which from the shift theorem is expressible as

$$\mathcal{F}\left(g(x-x')\right) = \mathcal{F}\left(g(x)\right) \cdot e^{-j2\pi s x'}$$

$$= G(s)\, e^{-j2\pi s x'}$$

(This follows from a substitution of variables).

Then

$$\mathcal{F}\left(f(x) * g(x)\right) = \int_{-\infty}^{\infty} G(s)\, e^{-j2\pi s x'} f(x')\, dx'$$

Since the integral only is over x',

$$\mathcal{F}\left(f(x) * g(x)\right) = G(s) \int_{-\infty}^{\infty} e^{-j2\pi s x'} f(x')\, dx'$$

or simply

$$\mathcal{F}\left(f(x) * g(x)\right) = G(s)\, F(s)$$

Hence $\quad f * g \supset G \cdot F$

Relation to filtering

So, how does this relate our two filtering operations? As an example, consider what happens when we filter a 1-D sequence using a rect() function in order to smooth it:

Assume we have a sequence $f(x)$. The filtered version will be

$$out(x) = f(x) * rect(x)$$

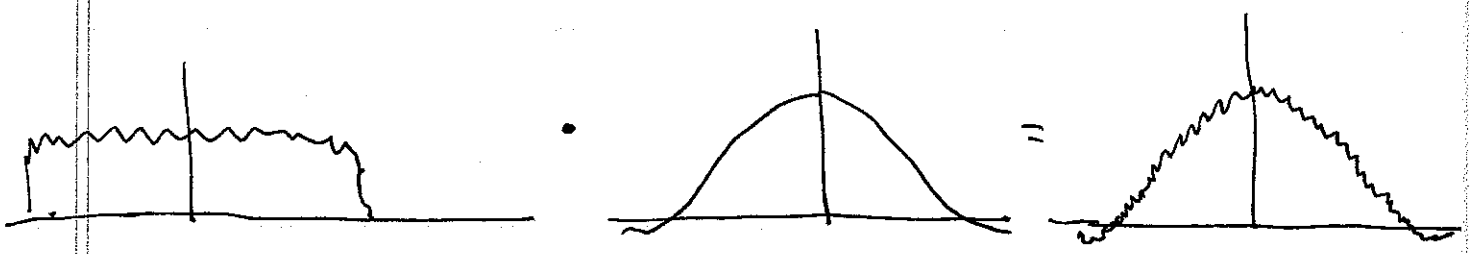
In the frequency domain,

$$Out(s) = \mathcal{F}\left(f(x) * rect(x) \right)$$

$$= F(s) \cdot sinc(s)$$

Note that we can think of the spectrum of the output image Out(s) as the input image spection F(s) weighted by sinc (s). This weighting depresses the high-frequency components of the image spectrum, leading to a blurring of the image.

Pictorially,



F(s)

So the filtered image is dominated by low-frequency components, as we expect in a smoothed image.

How about the other way around?

Time-domain view of low-pass filter

We also low-pass filtered images by multiplying spectra by rect() functions. In other words,

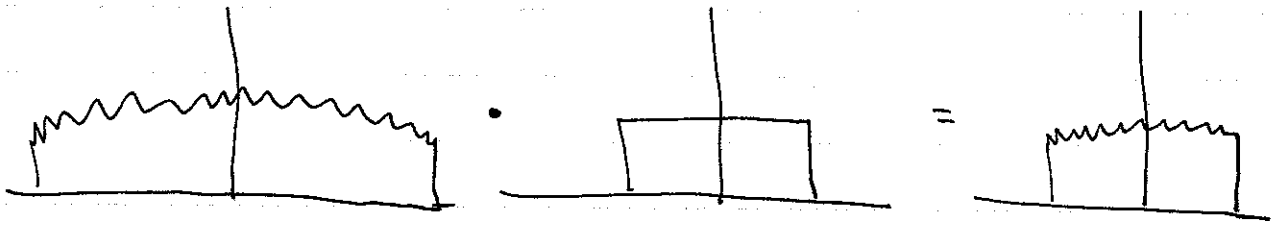
$$Out(s) = F(s) \cdot rect(s)$$

This tells us that the output image is the convolution of the input and a sinc() function:

$$out(x) = \mathcal{F}\left(F(s) \cdot rect(s) \right)$$

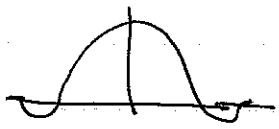
$$= f(x) * sinc(x)$$

Change the width of the rect in the frequency domain changes the width of the time-domain sinc function. Doubling the size of the rect causes the sinc to be half as wide, for example. This makes sense: a wider rect lets more frequency components pass through, and this corresponds to smoothing by a narrower sinc function.

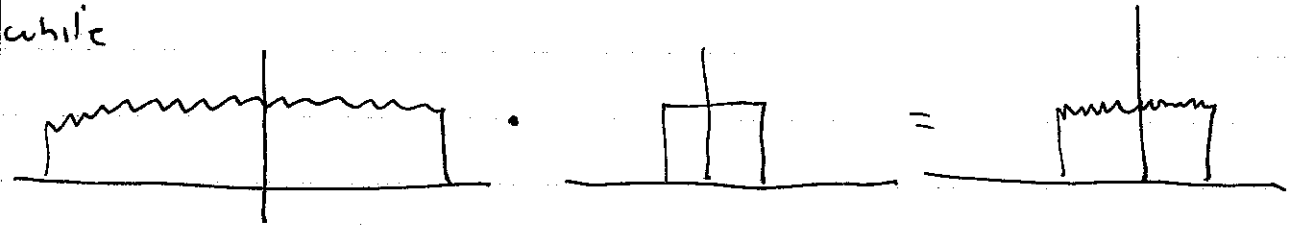


F(s)

is the same as convolution with



while



is the same as convolution with

Another example- Edge detection

Recall our discussion on edge detection, where we looked at several operators that can enhance edges in images. Now can we understand those as frequency-domain operations?

Consider a 1-D version of a Laplacian operator, which can be used to detect edges symmetrically, that is without regard to direction. One convolution operator to do this is

Laplacian : $[-1 \quad 2 \quad -1]$

which we can represent as $[-1 \quad 0 \quad -1] + [0 \quad 2 \quad 0]$, or in a continuous form as  , or

$$\text{Laplacian} = -\delta(x+1) + 2\delta(x) - \delta(x-1)$$

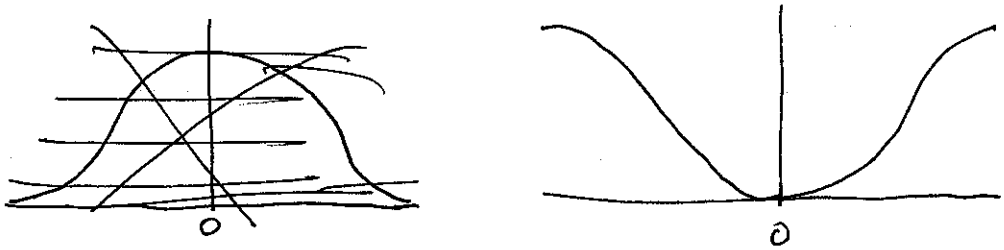
Convolution is equivalent to multiplication by the transform:

$$-\delta(x+1) + 2\delta(x) - \delta(x-1) \supset \; ?$$

Regrouping,

$$2\delta(x) - [\delta(x+1) + \delta(x-1)] \supset 2 - 2\cos 2\pi s$$

Hence in frequency space, the multiplicative weighting is plotted as



This function suppresses low spatial frequencies while passing high frequencies, as expected for an edge-enhancer.

This all generalizes easily to two dimensions, where the convolution theorem has the form

$$f(x,y) * g(x,y) \supset F(u,v) \cdot G(u,v)$$

Why do we need this equivalence?

It may seem wasteful to have two ways to do the same filtering operation. There are two main reasons we want to know this, however.

First, often we can visualize functions more easily in one domain than the other. Depending on whether we can describe an image or filter better in a given domain, we want to be able to visualize operations in either domain.

But, probably more importantly, it is usually much more efficient to compute large convolutions using frequency-domain operations that using time-domain operations. This is especially true if both f and g are comparable in size. This efficiency follows from the availability of the FFT algorithm.

Finally, for some types of problems, image formation is much easier using frequency domain approaches rather than time-domain approaches. A good example of this is tomography, such as CAT scanning in medical imaging.

How much more efficient?

Let's return to our problem of convolving two large images together. Let both f and g be nxn matrices. How many operations does it take to compute f*g in the time domain?

As before, when we talked about transform efficiency, for each output point, we have to multiply n^2 points by corresponding values in both matrices, so there are about n^4 operations needed to do this multiplication for every output location ($n^4 = n^2 \cdot n^2$).

For the frequency domain, here is our recipe for computing convolutions:

1) Transform $f(x,y)$ to $F(u,v)$
2) Transform $g(x,y)$ to $G(u,v)$
3) Multiply $G(u,v)$ times $F(u,v)$
4) Inverse transform the product to get solution

So three transforms are needed. Recall for a 2-D FFT we need $n^2 \log_2 n$ operations, so the entire convolution needs $3n^2 \log_2 n$. The ratio then for time to frequency domain is thus

$$\text{ratio} = \frac{n^4}{3n^2 \log_2 n} = \frac{n^2}{3 \log_2 n}$$

For a small image, with $n = 1000$, this is about

$$\text{ratio} = \frac{1000 \cdot 1000}{3 \cdot 10} \approx 3 \times 10^4$$

so it is far faster to do this in the time domain.

Correlation

Convolution has a cousin, called correlation, defined by the following integral:

$$f \star g = \int_{-\infty}^{\infty} f(x') \, g(x + x') \, dx'$$

↑ Star denotes correlation

↑ note plus sign

Correlation will prove to be very useful in signal detection, as the correlation between two signals will be maximum when they overlap exactly. This is most helpful when the signals are of zero mean, otherwise many cross terms make detection difficult.

There is a "correlation theorem" that corresponds to the convolution theorem:

$$f(x) \star g(x) \supset F^{*}(s)\, G(s)$$

This is the same as the convolution theorem except that we use the complex conjugate of the first function in the spectral multiplication.

We will see how this works in lab examples later.

Note that we have to be careful which function we conjugate and if we use the wrong one we will end up with an image that is backwards in the time/space domain. This will generally be obvious when we compute the correlation image.

A note on computing convolutions

When we calculate convolutions using the FFT, we in fact don't need to be as careful in calling the fftshift operation as we did in computing spectra. This is because both the two functions get mapped the same way into the frequency domain, and it doesn't really matter

how we represent frequencies. As long as we do element by element multiplications, we are ok.

However, if we have to make certain specific calculations on given frequency components, once again we must be attentive to how each coefficient is defined.