More on the Fourier transform

Bharath Hariharan

February 3, 2020

1 Summarizing the Fourier transform

The Fourier basis: The Fourier transform represents a change of basis for images. Instead of using the canonical basis where each basis element corresponds to a single pixel, we use the Fourier basis. For an $N \times N$ image, there are N^2 Fourier basis elements. The m, n-th pixel in the k, l-th basis image is given by:

$$B_{k,l}(m,n) = e^{i2\pi \frac{kx}{N} + i2\pi \frac{ly}{N}} \tag{1}$$

k determines the frequency along the horizontal and l denotes the frequency along the vertical. Figure **??** shows a few of the Fourier basis.

The Fourier transform: Given an $N \times N$ image $\mathbf{f}$, the Fourier transform gives us the representation of this image in the Fourier basis, $\mathbf{F}$.

$$F(k,l) = \sum_x \sum_y f(x,y) e^{-i2\pi \frac{kx}{N} - i2\pi \frac{ly}{N}} \tag{2}$$

Note that the Fourier transform is a linear transformation of the image. This should be the case since it is just a basis change.

The inverse fourier transform: Given the Fourier transform $\mathbf{F}$ of an image, we can get the image using the inverse Fourier transform, which simply combines the basis elements using the Fourier coefficients:

$$f(x,y) = \sum_k \sum_l F(k,l) B_{k,l}(x,y) = \sum_k \sum_l f(x,y) e^{i2\pi \frac{kx}{N} + i2\pi \frac{ly}{N}} \tag{3}$$

2 Convolution and the Fourier transform

With the Fourier transform, we now have two different representations of images: one in terms of the canonical basis and one in terms of the Fourier basis. For any operation we do on the one representation, there will be an equivalent operation that can be done on the other. Thus, we can perform operations on any representation. Borrowing from signal processing literature, when we perform operations on the canonical representation, we say that we are working in the "spatial domain". When we perform operations on the Fourier representation, we say that we are working in the "frequency domain".

Convolution and the Fourier transform: The two domains can also help us analyze how a particular operation affects the image. In particular let us look at convolution. Suppose we have an $n \times n$ image f. We want to convolve this with a $k \times k$ filter w. We want to understand how this operation affects the Fourier spectrum of f.

First, because the Fourier basis is dependent on the image size, we will need to bring both image and filter up to the same size. We will do this by padding with lots of zeros. In particular, we will pad f with $(k-1)/2$ zeros on all sides to get $\tilde{f}$ so that its size becomes $(n+k-1) \times (n+k-1)$. Similarly, we will pad w with $(n-1)/2$ zeros on all sides to get $\tilde{w}$ so that its size also becomes $(n+k-1) \times (n+k-1)$. We will now perform "same" convolution with this padded filter and image. It can be shown that the end result of this operation is the same as the output of a "full" convolution on the original image and filter. Denote this result by $\tilde{h}$.

Now let the Fourier transform of $\tilde{f}, \tilde{w}$ and $\tilde{h}$ be $\tilde{F}, \tilde{W}$ and $\tilde{H}$ respectively. Then the following fact is true for every k, l:

$$|\tilde{H}(k,l)| = |\tilde{F}(k,l)||\tilde{W}(k,l)| \tag{4}$$

In other words, *when we convolve two images in the spatial domain, we are multiplying together their amplitude spectra in the frequency domain.*

Therefore, to understand the impact of convolution with a particular filter w, we can construct its padded version $\tilde{w}$ as above and look at the amplitudes of the corresponding Fourier transform $\tilde{W}$. If there are frequencies (k,l) for which $|\tilde{W}(k,l)|$ is close to 0, then those frequencies will be *suppressed* in the output. Conversely, if $|\tilde{W}(k,l)| > 1$ those frequencies will be enhanced.

Gaussian blurring and the Fourier transform: As an example, let us consider convolution with Gaussian filters of varying standard deviation. Recall that as the standard deviation of the filter increases, the amount of blurring increases and we lose more details. Figure 1 shows Gaussian filters of increasing standard deviation (top row) and the corresponding amplitude spectrum (bottom row). As can be seen, the amplitude spectrum of a Gaussian filter *looks like a Gaussian*!

In addition we can make several observations, First observe that for all Gaussian filters, the amplitude of the Fourier coefficients is very low for high frequencies. Thus *all Gaussian filters suppress high frequencies.* Filters that *suppress high-frequencies while leaving low frequencies mostly intact* are called *low-pass filters.*

Second, note that while high frequencies are suppressed completely by the Gaussian filters, *all* frequencies (except the 0 frequency) are suppressed a little. Thus, the Gaussian filter does not exactly leave the low frequencies completely intact.

Third, observe that as the standard deviation of the Gaussian increases, more high frequencies are suppressed. For very high standard deviations, only very few low frequencies are preserved.

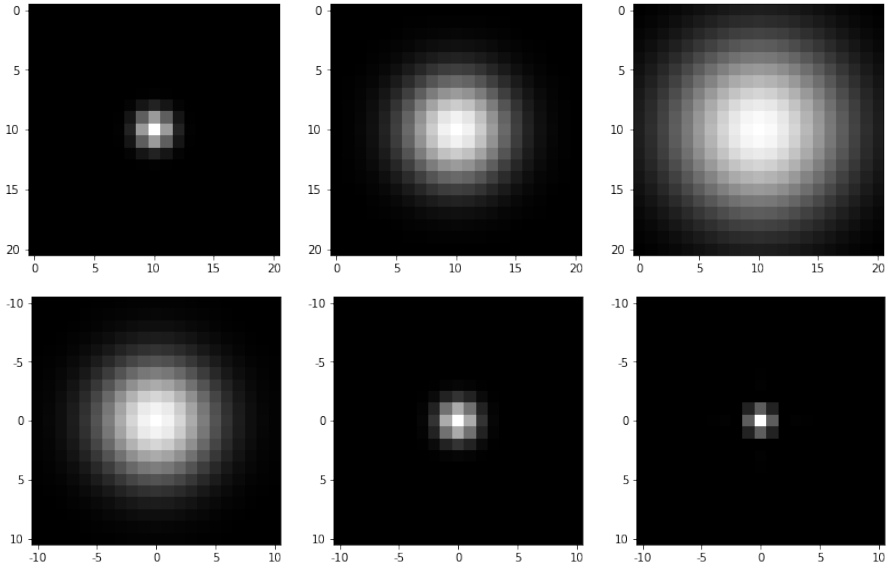

Figure 1: Top: Gaussian filters. Bottom: Amplitudes of the Fourier transform

Box filters: Figure 2 shows a corresponding visualization for the box filter (standard mean filter). Here, we have padded all box filters with zeros so that they are the same size. Similar arguments hold: all filters tend to suppress high frequencies. Interestingly, box filters do not do as clean a job as Gaussian filters in this regard: some high frequencies are not suppressed by the box filter. Thus a box filter is also a *low-pass filter*, but is less effective.

The ideal low-pass filter? Both the Gaussian filter and the mean filter are not ideal low-pass filters. The Gaussian filter tends to also downweight a few of the low frequencies. The box filter additionally fails to suppress some high frequencies.

What would be an ideal low-pass filter? In the frequency domain, this filter would have amplitude 1 for the frequencies lower than a threshold, and amplitude 0 for frequencies higher than a threshold. It turns out that this is incredibly hard to achieve. The closest one can get is the *sinc* filter:

$$w(x, y) = u(x)v(y) \tag{5}$$

$$u(x) = \frac{\sin(2\pi(x - x_0)/\sigma)}{2\pi(x - x_0)/\sigma} \tag{6}$$

$$v(y) = \frac{\sin(2\pi(y - y_0)/\sigma)}{2\pi(y - y_0)/\sigma} \tag{7}$$

Figure 3 shows this sinc filter of two different sizes. Unfortunately, small sized filters cannot actually achieve the effect we intend, because the sinc filter doesn't

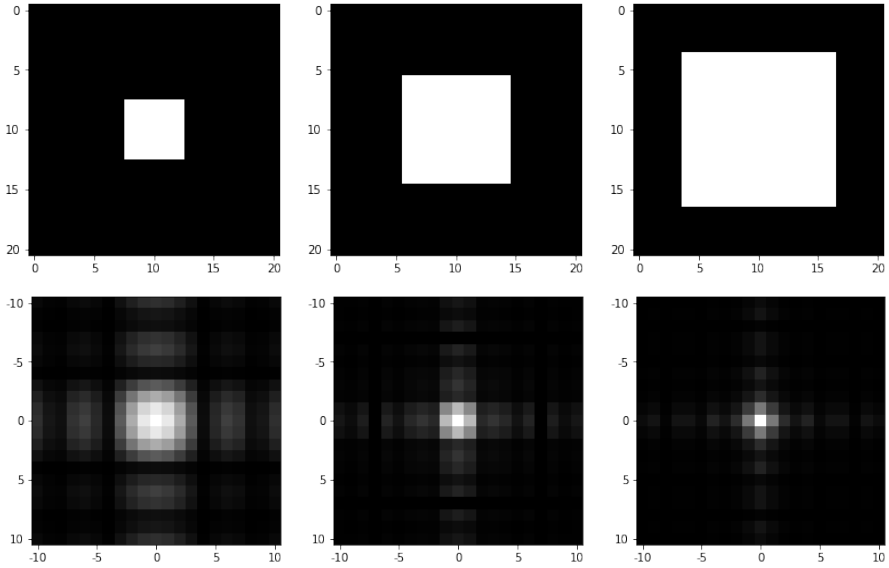

Figure 2: Top: Box filters. Bottom: Amplitudes of the Fourier transform

decay to 0 the way a Gaussian filter does. In fact, we need an *inifinitely large* kernel. As such the closest we can get to a *practical low-pass filter* is a Gaussian filter.

High-pass filtering What if we don't want to kill the low frequencies but the high frequencies instead? Figure 4 shows two filters. The first is the difference between the identity filter and a Gaussian filter. As can be seen this filter *suppresses* low frequencies and preserves high frequencies. This filter is thus a high-pass filter. Observe that to perform high-pass filtering, we don't need to specifically use this filter: we can simply subtract the low-pass filtered image from thee original.

More generally, a difference-of-Gaussian filter (right in Figure 4) will suppress both low and high frequencies and leave a *band* of frequencies in the middle in-tact. Such a filter is called a *band-pass filter*.

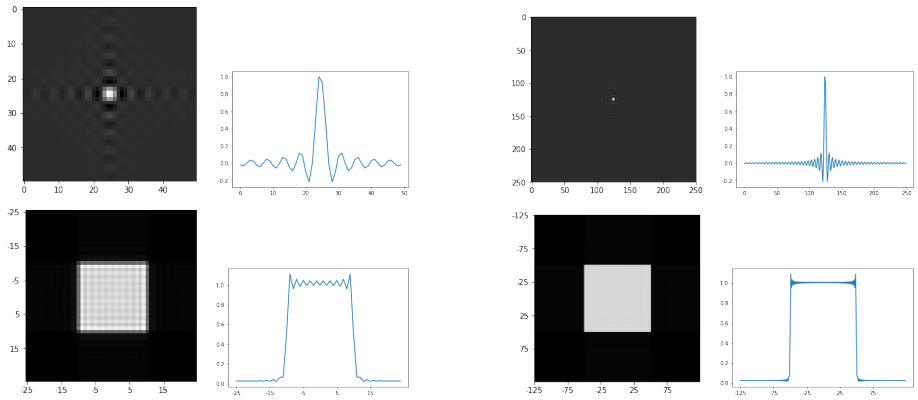

Figure 3: Top: Sinc filters, along with a plot of the intensity of the middle row. Bottom: Amplitudes of the Fourier transform, along with a plot of the amplitudes of the middle row.

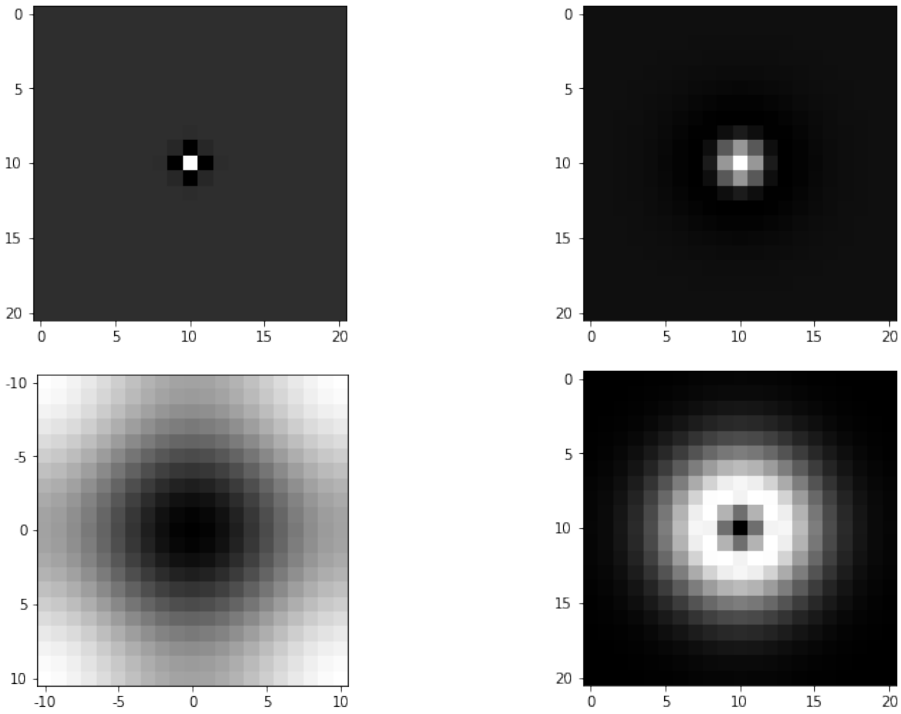

Figure 4: Top: The first filter shows the (identity-Gaussian) filter. Note that the visualization may be hard to see. The second filter is a difference of Gaussians. Bottom: Amplitudes of the Fourier transform