Image deconvolution

You've probably seen it on TV, in one of those crime drama shows. They have a blurry photo of a crime scene, and they click a few buttons on the computer and magically the photo becomes sharp and clear, so you can make out someone's face, or some lettering on a sign. Surely (like almost everything else on such TV shows) this is just science fiction? No. It's not. It's real and in this project, you'll write a program that does it.

When a photo is blurred each point on the photo gets smeared out according to some "smearing distribution," which is technically called a *point spread function*. We can represent this smearing mathematically as follows. For simplicity let's assume we're working with a black and white photograph, so that the picture can be represented by a single function $a(x, y)$ which tells you the brightness at each point (x, y). And let us denote the point spread function by $f(x, y)$. This means that a single bright dot at the origin ends up appearing as $f(x, y)$ instead. If $f(x, y)$ is a broad function then the picture is badly blurred. If it is a narrow peak, then the picture is relatively sharp.

In general the brightness $b(x, y)$ of the blurred photo at point (x, y) is given by

$$b(x,y) = \int_0^K \int_0^L a(x',y') f(x-x', y-y') \, dx' \, dy'.$$

where *KxL* is the dimension of the picture. This equation is called the *convolution* of the picture with the point spread function. Working with two-dimensional functions can get complicated, so to get the idea of how the math works, let's switch temporarily to a one-dimensional equivalent of our problem. Once we work out the details in 1D we'll return to the 2D version. The one-dimensional version of the convolution above would be

$$b(x) = \int_0^L a(x') f(x-x') \, dx'$$

The function $b(x)$ can be represented by a Fourier series

$$b(x) = \sum_{k=-\infty}^{\infty} \tilde{b}_k \exp\left(i\frac{2\pi k x}{L}\right),$$

$$\tilde{b}_k = \frac{1}{L}\int_0^L b(x) \exp\left(-i\frac{2\pi k x}{L}\right) dx$$

Where $\widetilde{b_k}$ are the Fourier coefficients. Substituting for $b(x)$ in this equation gives

$$\begin{aligned}\tilde{b}_k &= \frac{1}{L}\int_0^L \int_0^L a(x') f(x-x') \exp\left(-i\frac{2\pi k x}{L}\right) dx' \, dx \\ &= \frac{1}{L}\int_0^L \int_0^L a(x') f(x-x') \exp\left(-i\frac{2\pi k (x-x')}{L}\right) \exp\left(-i\frac{2\pi k x'}{L}\right) dx' \, dx.\end{aligned}$$

Now let us change variables to $X = x - x'$, and we get

$$\tilde{b}_k = \frac{1}{L}\int_0^L a(x') \exp\left(-i\frac{2\pi k x'}{L}\right) \int_{-x'}^{L-x'} f(X) \exp\left(-i\frac{2\pi k X}{L}\right) dX \, dx'$$

If we make $f(x)$ a periodic function in the standard fashion by repeating it infinitely many times to the left and right of the interval from 0 to L, then the second integral above can be written as

$$\int_{-x'}^{L-x'} f(X)\exp\left(-\mathrm{i}\frac{2\pi kX}{L}\right)\mathrm{d}X = \int_{-x'}^{0} f(X)\exp\left(-\mathrm{i}\frac{2\pi kX}{L}\right)\mathrm{d}X + \int_{0}^{L-x'} f(X)\exp\left(-\mathrm{i}\frac{2\pi kX}{L}\right)\mathrm{d}X$$

$$= \exp\left(\mathrm{i}\frac{2\pi kL}{L}\right)\int_{L-x'}^{L} f(X)\exp\left(-\mathrm{i}\frac{2\pi kX}{L}\right)\mathrm{d}X + \int_{0}^{L-x'} f(X)\exp\left(-\mathrm{i}\frac{2\pi kX}{L}\right)\mathrm{d}X$$

$$= \int_{0}^{L} f(X)\exp\left(-\mathrm{i}\frac{2\pi kX}{L}\right)\mathrm{d}X,$$

which is simply *L* times the Fourier transform $\widetilde{f_k}$ of *f* (*x*). Substituting this result back into our equation for $\widetilde{b_k}$ we then get

$$\tilde{b}_k = \int_0^L a(x')\exp\left(-\mathrm{i}\frac{2\pi kx'}{L}\right)\tilde{f}_k\,\mathrm{d}x' = L\,\tilde{a}_k\tilde{f}_k.$$

In other words, apart from the factor of *L*, the Fourier transform of the blurred photo is the product of the Fourier transforms of the unblurred photo and the point spread function. Now it is clear how we deblur our picture. We take the blurred picture and Fourier transform it to get $\widetilde{b_k} = L\,\widetilde{a_k}\widetilde{f_k}$. We also take the point spread function and Fourier transform it to get $\widetilde{f_k}$. Then we divide one by the other:

$$\frac{\tilde{b}_k}{L\tilde{f}_k} = \tilde{a}_k$$

which gives us the Fourier transform of the *unblurred* picture. Then, finally, we do an inverse Fourier transform on $\widetilde{a_k}$ to get back the unblurred picture. This process of recovering the unblurred picture is called *deconvolution*. Real pictures are two-dimensional, but the mathematics follows through the same. For a picture of dimensions *KxL* we find that the two-dimensional Fourier transforms are related by:

$$\tilde{b}_{kl} = KL\tilde{a}_{kl}\tilde{f}_{kl}$$

and again, we just divide the blurred Fourier transform by the Fourier transform of the point spread function to get the Fourier transform of the unblurred picture. In the digital realm of computers, pictures are not pure functions *f*(*x*, *y*) but rather grids of samples, and our Fourier transforms are discrete transforms not continuous ones. But the math works out the same again. The main complication with deblurring in practice is that we don't usually know the point spread function. Typically, we have to experiment with different ones until we find something that works. For many cameras it's a reasonable approximation to assume the point spread function is Gaussian:

$$f(x,y) = \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

where σ is the width of the Gaussian. Even with this assumption, however, we still don't know the value of σ and we may have to experiment to find a value that works well. In the following exercise, for simplicity, we'll assume we know the value of σ.

a) You will find a file called blur.txt that contains a grid of values representing brightness on a black-and-white photo—a badly out-of-focus one that has been deliberately blurred using a Gaussian point spread function of width $\sigma = 25$. **Write a program that reads the grid of values into a two-dimensional array of real numbers and then draws the values on the screen of the computer as a density plot.** You should see the photo appear. If you get something wrong, it might be upside-down. Work with the details of your program until you get it appearing correctly. (Hint: The picture has the sky, which is bright, at the top and the ground, which is dark, at the bottom.)

b) Write another program that **creates an array, of the same size as the photo, containing a grid of samples drawn from the Gaussian** $f(x, y)$ above with $\sigma = 25$. **Make a density plot of these values on the screen too, so that you get a visualization of your point spread function**. Remember that the point spread function is periodic (along both axes), which means that the values for negative x and y are repeated at the end of the interval. Since the Gaussian is centered on the origin, this means there should be bright patches in each of the four corners of your picture, something like this:

c) Combine your two programs and add Fourier transforms using the functions rfft2 and irfft2 from numpy.fft, to make a program that does the following:

i) Reads in the blurred photo
ii) Calculates the point spread function
iii) Fourier transforms both
iv) Divides one by the other
v) Performs an inverse transform to get the unblurred photo
vi) Displays the unblurred photo on the screen

When you are done, you should be able to make out the scene in the photo, although probably it will still not be perfectly sharp. Hint: One thing you'll need to deal with is what happens when the Fourier transform of the point spread function is zero, or close to zero. In that case if you divide by it, you'll get an error (because you can't divide by zero) or just a very large number (because you're dividing by something small). A workable compromise is that if a value in the Fourier transform of the point spread function is smaller than a certain amount (e.g. 10^{-3}) you don't divide by it—just leave that coefficient alone.

d) Bearing in mind this last point about zeros in the Fourier transform, what is it that limits our ability to deblur a photo? Why can we not perfectly unblur any photo and make it completely sharp?