Blind Image Deconvolution

Aron Liu

June 1, 2014

Abstract

When images are taken, the image itself is just a representation of what you actually see. The camera's image is susceptible to motion blur, bad reflections, and the camera's own limitations (e.g. how many megapixels). Having a camera image be as close as possible to the true image is advantageous in many applications across all different fields, e.g., photography, astronomy, physics, etc. You can think of a camera image as the true image combined with some "blurring factor". One way to get back the true image is by deconvolution, a method that separates the camera image from its "blurring factor".

This project will focus on some rudimentary issues with image deconvolution: accounting for random noise and what to do when the "blurring factor" is unknown, i.e., blind image deconvolution.

1 The Basic Problem

Image deconvolution is a prototypical example of an inverse problem, i.e., a problem where you want to find the inputs (the true image and the "blurring factor") given the outputs (the camera image). One issue with inverse problems is they can be extremely sensitive to the inputs. If one of the inputs changes slightly, the output can be completely different.

Definition A problem is **ill-posed** if one of the following is true:

1. a solution does not exist
2. a solution is not unique
3. the solution's behavior does not change continuously with the inputs

In many cases, the "blurring factor", also known as the **Point Spread Function** (**PSF**), takes the form of a large and unwieldy matrix with one value for each pixel. Such an ill-conditioned matrix will almost always cause image deconvolution to be an ill-posed inverse problem. A consequence of this is if there is any noise unaccounted for in the PSF, the deconvolved image will generally not be a great representation of the true image.

Notation note: Throughout this write-up, the output image or blurred image will be referred to as the camera image. The exact image will be referred to as the true image. The "blurring factor" will be referred to as the PSF.

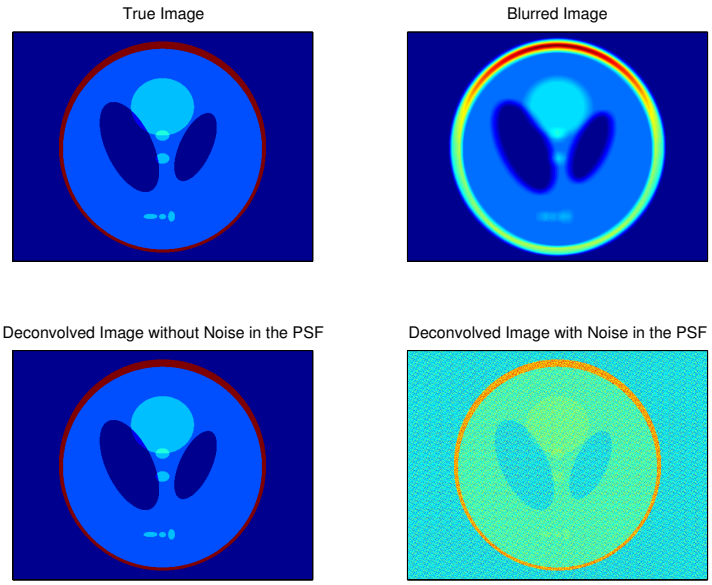

Figure 1: An example of how noise can overwhelm a deconvolved image

2 Regularization

One way to solve ill-posed inverse problems is by a process called **regularization**. This process introduces additional information in the form of extra constraints.

2.1 Least Squares Method

The regularization method we will be using builds off the least squares method for solving a system. Given a problem

$$Ax = b$$

where A is a matrix, x is the input vector, and b is the output vector, we want to find the optimal x given A and b. One solution to this problem is to find x such that it minimizes the squared norm of the difference of the two sides of the equation, i.e.,

$$\min_x \|Ax - b\|_2^2$$

In the context of image deconvolution, we can think of A as the PSF, x as the true image, and b as the camera image. However, if b has a bit of noise in it, i.e.,

$$b = \hat{b} + \varepsilon$$

for ε some noise and $\hat{b}$ the actual output we want to find an optimal x for, we can find a bad solution if the problem is ill-posed. Remember, the intuition behind

an ill-posed problem is even if b and $\hat{b}$ are extremely similar to each other, the optimal x for each can look completely different.

2.2 Tikhonov Regularization

Tikhonov regularization, named for Russian mathematician Andrey Tikhonov, attempts to fix the issue that arises when the least squares method is used with an ill-posed inverse problem by adding an additional constraint on the minimization:

$$\min_x \{\|Ax - b\|_2^2 + \lambda^2 \|x\|_2^2\}$$

In the new minimization, the additional constraint is the squared norm of x. λ is a non-negative constant decided in advance, acting as a weight on the strength of the additional constraint. Some alternate ways to write Tikhonov regularization include

$$(A^T A + \lambda^2 I)x = A^T b$$

and

$$\min \left\| \begin{bmatrix} A \\ \lambda I \end{bmatrix} x - \begin{bmatrix} b \\ 0 \end{bmatrix} \right\|$$

Intuitively, why does this help solve an ill-posed problem? When deconvolving an image, we divide out the PSF's coefficients from the camera image's coefficients in the Fourier domain, i.e.,

$$\tilde{f}_m = \frac{\tilde{h}_m}{\tilde{g}_m} + \frac{\tilde{\varepsilon}_m}{\tilde{g}_m}$$

where $\tilde{f}$ represents the true image's Fourier coefficients, $\tilde{h}$ represents the camera image's Fourier coefficients, and $\tilde{g}$ represents the PSF's Fourier coefficients. ε represents the noise or error in h. When the PSF's coefficients are small, dividing by them amplifies the noise term $\frac{\tilde{\varepsilon}_m}{\tilde{g}_m}$ in the camera image's coefficients.

The additional constraint that Tikhonov regularization adds is the "size" of x, meaning it favors small coefficients over large ones. Basically, it prevents pixel values of x from blowing up to extreme values and amplifying any noise. It can be thought of as a dampening of the coefficients in the Fourier domain that have large error from the noise in the camera image.

In a mathematical sense, the additional constraint acts as a weight on each coefficient in the Fourier domain. For the PSF's Fourier coefficients that are sufficiently small, the corresponding true image's Fourier coefficients are multiplied by a weight close to zero, since these coefficients are overwhelmed by error from noise. Figure 2 shows how this weight changes based on the PSF's Fourier coefficients. There is a threshold at $\sqrt{\lambda}$, derived from the minimization in the Fourier domain.

For choice of λ, there is an optimal value specific to each problem that can be found through heuristic methods. If λ is too small, the amplified noise won't be dampened enough. If λ is too large, the image will be dampened too much and become a poor representation of the true image. Figure 3 shows an example of this.

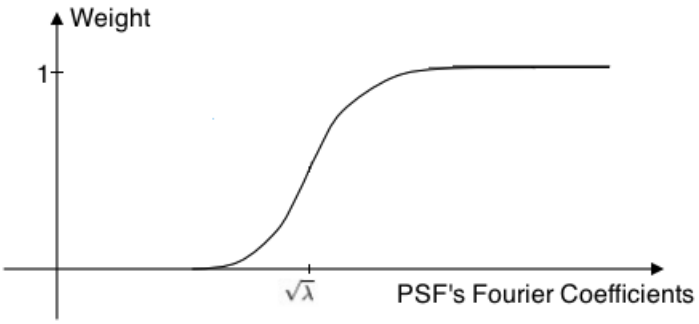

Figure 2: How Tikhonov regularization weights Fourier coefficients. There is a threshold $\sqrt{\lambda}$ below which the weight smoothly goes to zero

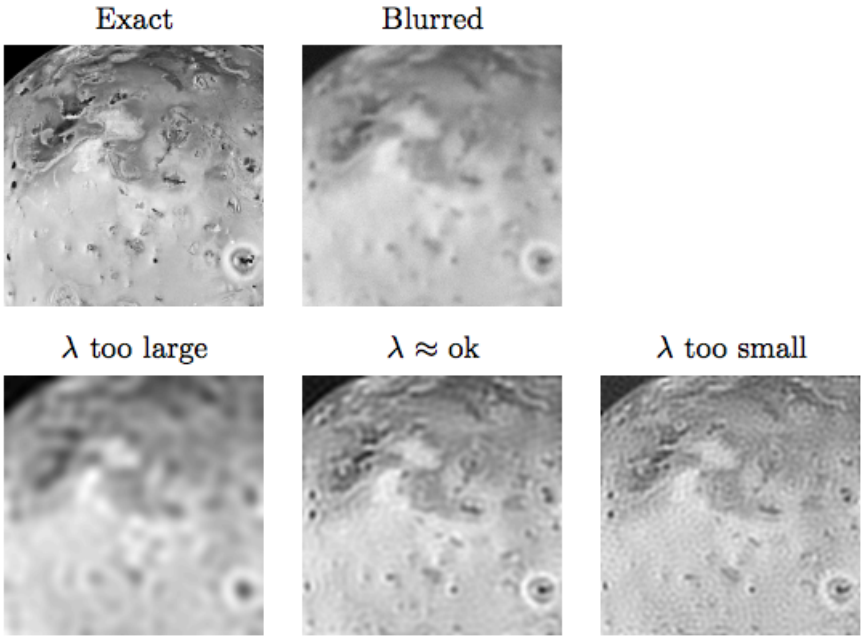

Figure 3: λ does have an optimal value that produces the best approximation of the true image

3 Blind Image Deconvolution

Tikhonov regularization is a good method for approximating the true image when the PSF is known, but what about when the PSF isn't known? Most real life applications of image deconvolution need to be able to deconvolve an image without knowing the exact PSF.

Definition Attempting to reconstruct the true image *and* the PSF from a camera image is called **blind image deconvolution**.

There are many algorithms that attempt to solve blind image deconvolution (enough to fill entire textbooks just on this problem!). Here we will focus on one of these algorithms, iterative blind image deconvolution.

3.1 Iterative Blind Image Deconvolution

Definition An **iterative algorithm** is one that takes an initial input, approximates the output, then takes this approximation as an input and repeats to create a sequence of approximations that converge toward the true solution.

There are many iterative algorithms for solving blind image deconvolution, because at each iteration you can do many different things to get the next approximation. In our examples, we will use basic image deconvolution with Tikhonov regularization at each iteration.

However, in order to get the next iteration, we have to have some prior knowledge about either the PSF or the true image (we can't find the true image if we know zero information about how it got altered!). At each iteration, we apply this knowledge to the outputs of the previous iteration before creating the new outputs.

For example, if we know that the PSF or the true image has all non-negative values, we can simply take all negative values in the previous output and set them to 0 before proceeding with the algorithm. Other examples of prior knowledge include knowing the pixel dimensions of the PSF, knowing a bandwidth of values the PSF or true image must stay between, or knowing something about the kind of effect the PSF had on the true image (e.g. motion blur, radial blur, etc.). Figure 4 summarizes how iterative blind deconvolution works, while Figures 5-7 show an example of iterative blind deconvolution in use.

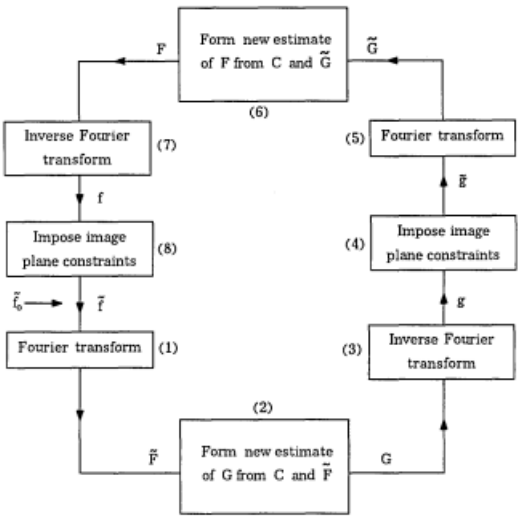

Fig. 1. General deconvolution algorithm.

Figure 4: Given $\tilde{f}_0$ as an initial guess for the true image and C the camera image, steps (1), (2), and (3) use the Fourier domain to find an initial guess for g, the PSF. Step (4) is where prior knowledge is applied to change the PSF. From the changed PSF, steps (5), (6), and (7) find a new approximation of the true image. Step (8) is again where prior knowledge is applied to change the true image. Then the algorithm repeats.

3.2 Complexity

The downside of iterative algorithms is their computing time. In our algorithm, each step uses Tikhonov regularization. Tikhonov regularization is optimized at complexity $O(N \log^2 N)$. The number of steps depends on what prior knowledge you have. The convergence rate can be quite slow, meaning the true time complexity can be much worse than $O(N \log^2 N)$.

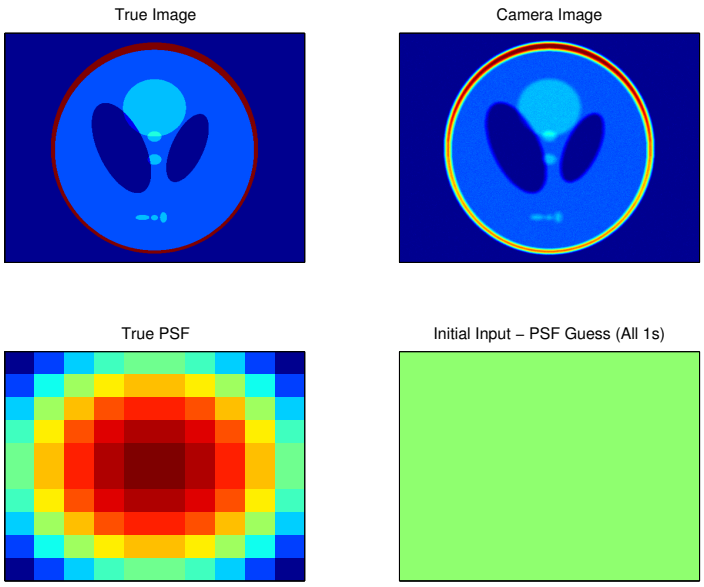

Figure 5: An example of iterative blind deconvolution. The initial input was a guess at the PSF, simply all 1s. Conditions on the PSF were its 10x10 dimensions and non-negative entry values. Figures 6 and 7 show the results.

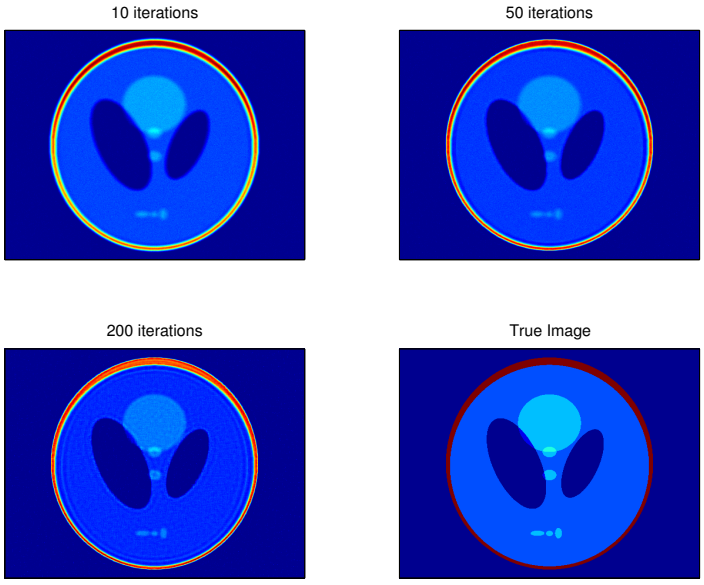

Figure 6: The approximated true images

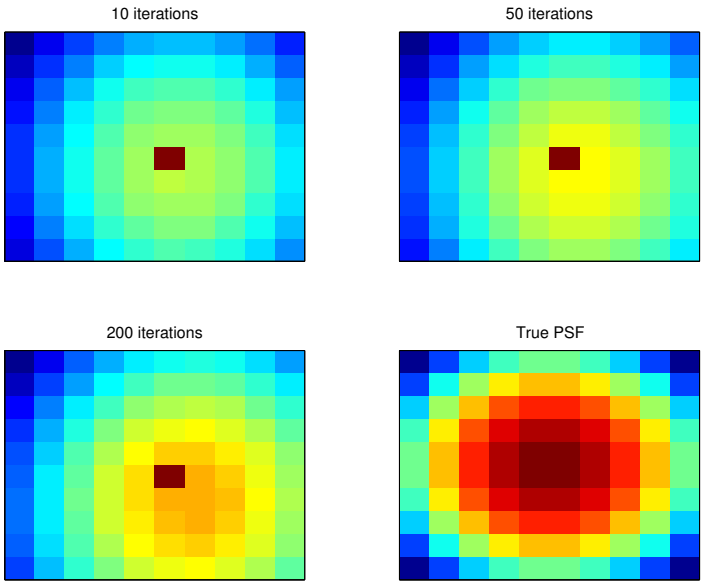

Figure 7: The approximated true PSFs

4 References

http://www.sciencedirect.com/science/article/pii/S0377042706003839
http://www.deconvolve.net/DNResults2D.html
http://tesla.desy.de/~rasmus/dr/bibliography/pdf/biggs97a.pdf
http://webdocs.cs.ualberta.ca/~nray1/CMPUT615_2010/Smoothing/Regularization.pdf
http://www.imm.dtu.dk/~pcha/DIP/chap4.pdf
http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV0405/UMAR/AVassign2.pdf
http://www.lx.it.pt/~bioucas/IP/
http://gerard.meurant.pagesperso-orange.fr/p_plzen_2010.pdf