Image Deblurring and Noise Reduction in Python

Vincent DeVito
Computer Systems Lab
2009-2010

Abstract

In the world of photography and machine vision, blurry images can spell disaster. They can ruin an otherwise perfect photo or make it impossible for a computer to recognize the image or certain components of it for processing. The best way to counter this without taking another, clearer picture is to utilize deconvolution techniques to remove as much blur as possible.

My plan was to first design a program that takes an image, blurs it using a known blur kernel, then deblurs it to reproduce the original image. Then I would design a noise reduction algorithm to reduce the noise in the output of the deconvolution algorithm.

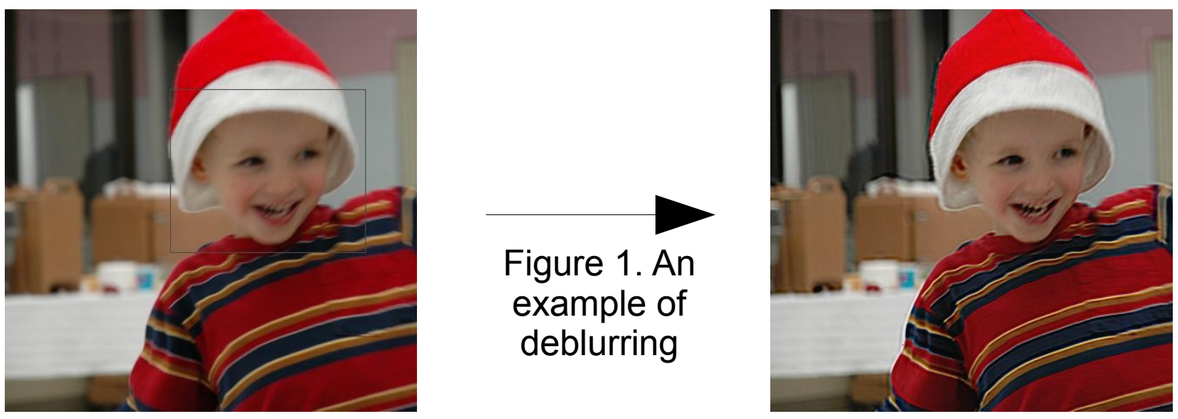

Figure 1. An example of deblurring

Methods

The convolution and deconvolution process heavily rely upon the Fourier transform (Figures 2 & 3). The 2D Fourier transform converts images from the spatial domain to the frequency domain with complex values. This makes convolution and deconvolution simple, since they are just a matter of point multiplication or division, respectively, of the transformed image's pixel values with the transformed blur kernel's pixel values. From there, the inverse Fourier transform (Figure 4) converts the convolution/deconvoluted image back to the spatial domain.

$$F(x,y) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n) e^{-j2\pi\left(x\frac{m}{M}+y\frac{n}{N}\right)}$$

Figure 2. 2D Fourier Transform

$$P(k,b) = \frac{1}{N}\sum_{a=0}^{N-1} f(a,b)\, e^{-\iota 2\pi\frac{ka}{N}}$$

$$F(k,l) = \frac{1}{N}\sum_{b=0}^{N-1} P(k,b)\, e^{-\iota 2\pi\frac{lb}{N}}$$

Figure 3. Separated Fast Fourier Transform

$$f(m,n) = \frac{1}{MN}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} F(x,y) e^{j2\pi\left(x\frac{m}{M}+y\frac{n}{N}\right)}$$

Figure 4. 2D Inverse Fourier Transform

For my project, I can only use motion blur kernels, since my method of image division deconvolution does not work with other kernels and will only produce a noisy, indecipherable image. Also, for my noise reduction program, I designed my own algorithm that first selectively Gaussian blurs using only pixels deemed not noise based on an average of the surrounding pixels. Then I applied an unsharp sharpening filter to restore edge clarity.

Background

In my research I have found various methods of blind and non-blind image deconvolution. One paper discussed comparing a blurry, correct intensity image with a sharp, noisy image to produce a proper, deblurred output image with few artifacts. Another paper discussed an algorithm they developed to estimate the blur kernel and use that to deblur the image from just a single photograph. Various deconvolution algorithms already exist, and it is the other component, the blur kernel, that requires further research. The more accurately the blur kernel can be estimated, the more accurate and clear the output image will be. For noise reduction, I found a paper that discusses using edge detection and modified Gaussian smoothing kernels to smooth the image while maintaining edge sharpness.

Results

I completed my project to the extent that I originally expected. My convolution programs works correctly and can convolute any image using any blur kernel. My deconvolution program works correctly as long as the kernel is known and is a motion blur kernel. The output, however, is noisy. Fortunately, my noise reduction program works very well, arguably surpassing the abilities of the aforementioned published noise reduction algorithm I found in my research. My results are outlined in the figures below.

For future work, I would research more into blur kernels and point spread functions to better understand the deconvolution process and make it more adaptive. The next logical step is to create a program to attempt and estimate the blur kernel from a naturally blurred image so that this process can be used in real world application. Finally, as can be seen in Figure 6, the final output image has different contrast and image intensity than the original input image. A possibility for future work would be to try and add an algorithm to increase contrast and correct the image intensities to better match the original, desired image.

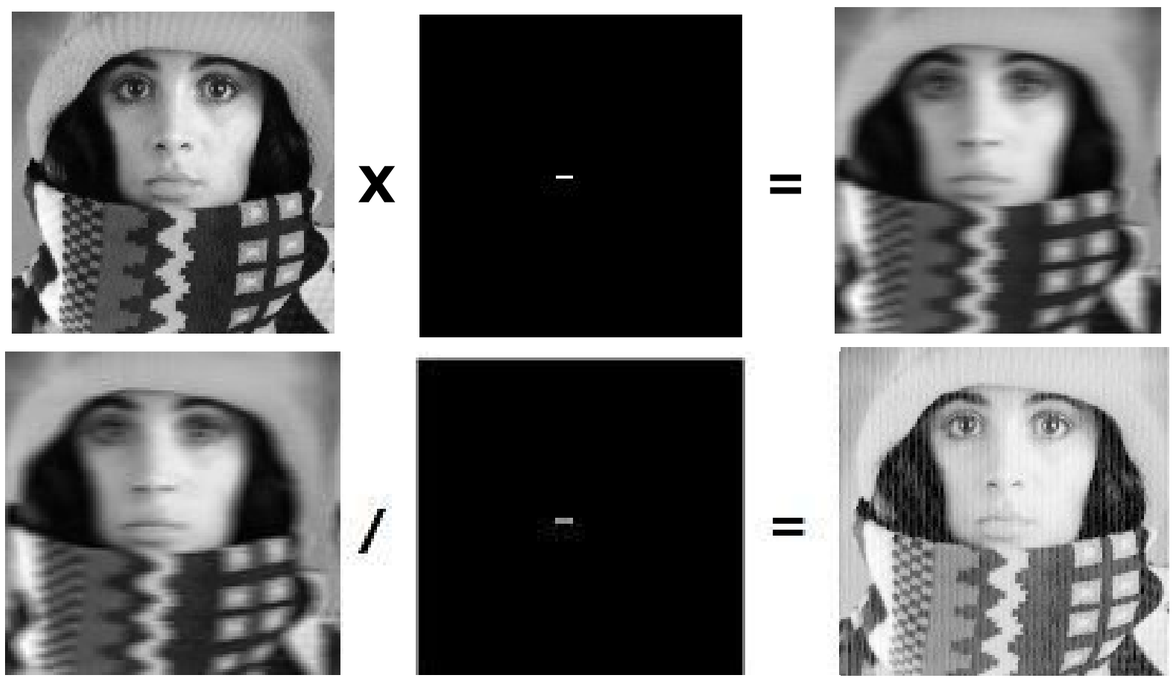

Figure 5. A picture being blurred using the given kernels, then being deblurred using that same kernel.

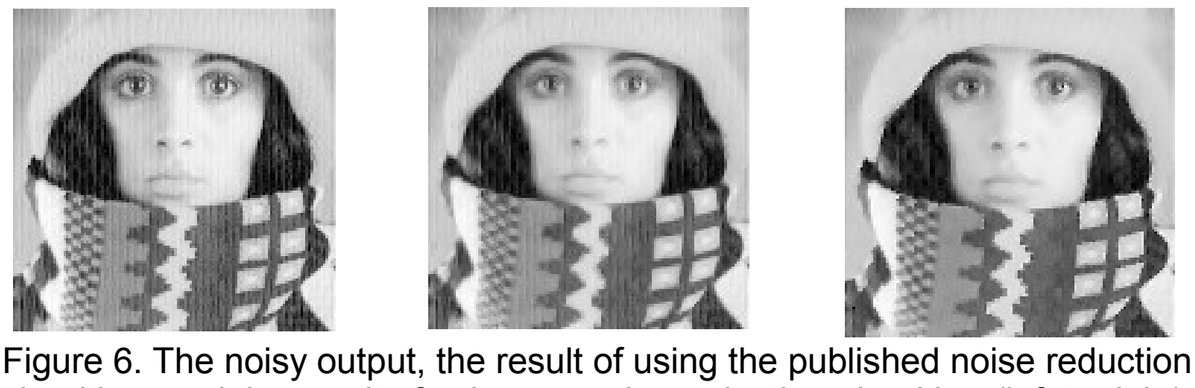

Figure 6. The noisy output, the result of using the published noise reduction algorithm, and the result of using my noise reduction algorithm (left to right).