

Image Deblurring Techniques

TJHSST Senior Research Project Proposal

Computer Systems Lab 2009-2010

Vincent DeVito

January 29, 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. That is the design of this project. My plan is to first design a program that takes an image, blurs it using a known blur kernel, then deblurs it to reproduce the original image. After that I will attempt to create a program to determine the blur kernel of a naturally blurred image. I will use Python and a package called Python Imaging Library that will allow me to utilize multiple image formats. My success will be measured simply by how much the output (deblurred) image matches the input (original) image.

Keywords: deblurring, deconvolution, image processing, noise reduction

1 Introduction and Background

The goal of this project is to create a program that can take an image input that has been blurred (first artificially, and later hopefully by poor image capture) and to employ image deblurring techniques to restore the image and create a sharp, more recognizable output image with as few blur artifacts and noise as possible.

1.1 Previous Research

So far I have found a paper regarding image deblurring and noise suppression called "Image Deblurring with Blurred/Noisy Image Pairs" by Lu Yuan, et al.² that I plan to utilize in helping me understand the techniques and algorithms that go into reducing the noise of and deblurring an image. In their research they used a blurry image with proper intensity and poor sharpness and paired it with an identical picture with good sharpness but poor intensity and riddled with noise to create a sharp, correct intensity output with few or no artifacts left in the output image.

Another paper¹ I have read discusses an algorithm that the group of researchers discovered that allows for a mostly accurate estimation of the blur kernel, or the path/direction of the blur from the original im-

age. Their algorithm takes four inputs: the blurry image, a section of the image that has a good sample of blurring (in case the image is not uniformly blurred), if the blur is estimated to be more horizontal or more vertical, and the estimated size of the blur kernel. Given these inputs, their algorithm can sufficiently estimate the blur kernel such that the image, which was captured using poor technique with a standard, off-the-shelf camera, is satisfactorily deblurred with few artifacts after deconvolution. Any artifacts that are left can generally be removed by an experienced photo editor.

1.2 Other Research

Through my own work I have accrued a detailed understanding of basic and intermediate image processing techniques and algorithms from various online worksheets and lessons at <http://homepages.inf.ed.ac.uk/rbf/HIPR2/wksheets.htm>. I plan to use these techniques to help me code and understand the more complex concepts behind image deblurring and the intermediate steps involved. For example, I have extensively used the section referring to the Fourier Transform, located here: <http://homepages.inf.ed.ac.uk/rbf/HIPR2/fourier.htm>.

2 Development

2.1 Project Design

I used the programming language Python to write the code for this project. I decided to use Python because of its simplicity and adaptability. I also found a package online called the Python Imaging Library (PIL) that allows me to read in and process images in almost any format using Python with relative ease. This enables me to use more compressed image formats, such as .jpg and .gif, instead of the larger uncompressed formats such as .ppm and .pgm as well as letting me process them much more efficiently. However, throughout my testing I continue to use both .gifs and .pgms so that I can confirm that accuracy of PIL as well as troubleshoot my code with greater ease and efficiency.

The first step in this project is to create a program that can artificially blur an input image using a known and given blur kernel. This is accomplished by converting both images to the frequency domain, using the Fast Fourier Transform (FFT), doing

a simple point multiplication of the images, then converting them back to the spatial domain using the Inverse Fast Fourier Transform (IFFT).

The next step is to write the deconvolution algorithm that, when given an image and its known blur kernel, can deblur the input image. This is fairly straightforward and will likely involve the reverse of the aforementioned "blurring" algorithm. After this step, I can attempt to add a noise reduction filter to remove any excess noise in the image and further sharpen and clarify the image.

The final step is to design a program that can estimate the blur kernel. This program will first be tested on blurry images with known blur kernels to make sure that the estimated blur kernel is similar to the actual blur kernel used. Then, after deemed acceptable, this program will estimate the blur kernel of a naturally blurred image with an unknown blur kernel, which can then be used to deblurring that image, hopefully to an acceptable level. This last step is a large one and currently under a lot of research since blind deconvolution, as it is known, is quite difficult.

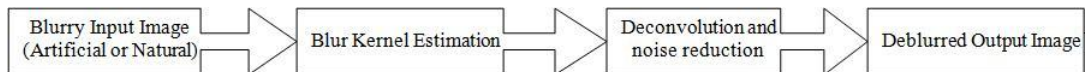


Figure 1. A graphic representation of my program design.

2.2 Testing

My project's success will be measured by its ability to take an image (which will most likely be artificially blurred) and return it to its original, sharp quality. I will test my projects adaptability and thoroughness by running a series of tests that will entail attempting to deblur images of different con-

tent, type, and contrast with varying magnitudes of blur. This will test my programs ability to repair images regardless of content or magnitude of blur distortion, although there will obviously be an upper limit to the amount of distortion that can plausibly be removed. An example of a successful run is illustrated below:



Figure 2. This is an example of a blurry image input, with a particularly blurry section highlighted.¹



Figure 3. This is the same section from figure 2, but with the blur drastically reduced, due to deconvolution.¹

2.3 Theory

2.3.1 Fourier Transform

The Fourier Transform is heavily involved with image convolution and deconvolution because it allows for greater speed and simpler code. The Fourier Transform converts values in an array from the spatial domain to the frequency domain using a sum of complex numbers, as given by the

$$F(x) = \sum_{n=0}^{N-1} f(n) e^{-j2\pi(x\frac{n}{N})}$$

equation:

The 2-Dimensional Discrete Fourier Transform (DFT) does this using a matrix or 2D array of values and uses a nested sum:

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

Since the 2-Dimensional Discrete Fourier Transform uses a nested sum, it can be separated to create two 1-Dimensional Fourier Transforms in a row, first in one direction (vertically or horizontally), then in the other direction.

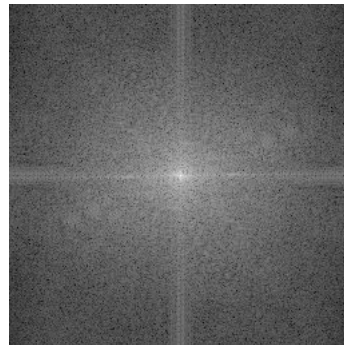
$$P(k, b) = \frac{1}{N} \sum_{a=0}^{N-1} f(a, b) e^{-i2\pi\frac{ka}{N}}$$

$$F(k, l) = \frac{1}{N} \sum_{b=0}^{N-1} P(k, b) e^{-i2\pi\frac{lb}{N}}$$

This is known as the Fast Fourier Transform (FFT) and runs significantly faster than the DFT, since the DFT has a runtime of $O(n^2)$ and the FFT has a runtime of $O(n \log_2 n)$. The following is an example of a picture being converted the spatial domain to the frequency domain via the Fourier Transform.



is then transformed to



The reason the FFT is so important to image convolution and deconvolution is that it takes long iterative algorithms and turns them into

simple point arithmetic. For example, image convolution becomes as simple as taking the Fourier Transform of the image and the blur kernel (known as the Point Spread Function (PSF) after transformation), transforming them to the frequency domain and point multiplying the two images. Then the two images can be converted back to the spatial domain by the Inverse Fourier Transform, given by

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

and the result will be a blurry (convoluted) image. To reverse this process and deconvolute the image, assuming the blur kernel is known, is as simple as point dividing the transformed image by the PSF, instead of

multiplying.

3 Results

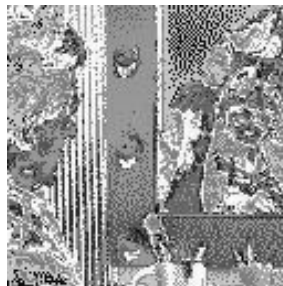
I have successfully transformed an image and then inversely transformed it to produce an almost identical output image using my program. I have not yet successfully blurred an image, however.

3.1 Errors

My program has not been entirely debugged and has some running errors. These have manifested themselves during the transform and inverse transform. Some images are inverted in color and some images are distorted or pixelated. For example:



is output as



I do not know why the errors are not consistent, but I speculate that the imaging library I'm using might be to blame. To test this, I plan to switch to using .pgm files, since they are uncompressed and do not require an imaging library to read in Python.

3.2 Scope

The scope of this project is rather narrow, but important. It pertains only to blurry images, but this is a rather large problem in the worlds of image processing, photography, and machine vision. In photogra-

phy, blurry images are undesirable because they lack sharpness or clarity and in machine vision, blurriness can make an image indecipherable by the computer or render certain processes ineffective, such as edge detection. Blind image deconvolution is also a very large area of research, specifically on the subject of estimating unknown blur kernels to increase adaptability for deblurring programs to deblur any image.

4 Conclusions

I have managed to transform and inversely transform an image using the Fourier Transform with partial success. These processes will be crucial to performing image convolution and deconvolution as the next step in my project.

4.1 Future Work

There is a lot of room for future work on this project since, at this point, I

only have the FFT and IFFT working with inconsistent success. The first and most obvious area for improvement is to troubleshoot my FFT and IFFT code so that it is successful and reliable. Then, the next step is to blur an image and experiment with different kinds of blur kernels. From there comes the obvious step of reversing the blur using that known kernel and experimenting with data loss and noise reduction on the output images. The final area of research for this project is the one that is most applicable to the real world and also the subject of much study in the computer science community. This is the area of blind deconvolution, which estimates the blur kernel from an image in which it is not known and then deconvolutes the image based on this estimate. Research is ongoing in trying to find the most efficient and adaptive method of estimating the blur kernel.

References

- [1] Brayer, J. M. (n.d.). Introduction to the Fourier transform. In *Topics in human and computer vision*. Retrieved from University of New Mexico Department of Computer Science website: <http://www.cs.unm.edu/brayer/vision/fourier.html>
- [2] Fergus, R., Singh, B., Hertzmann, A., Roweis, S. T., & Freeman, W. T. (2006, July). Removing camera shake from a single photograph. *ACM*

Transactions on Graphics, 25(3), 787-794. doi:10.1145/1141911.1141956

- [3] Fisher, R., Perkins, S., Walker, A., & Wolfart, E. (2000, October). *Hypermedia Image Processing Resource (HIPR2)* [Image processing learning resource]. Retrieved from <http://homepages.inf.ed.ac.uk/rbf/HIPR2/wksheets.htm>
- [4] Smith, S. W. (1997). A closer look at image convolution. In *The scientist and engineer's guide to digital signal processing* (pp. 418-422). Retrieved from <http://www.dspguide.com/>
- [5] Yuan, L., Sun, J., Quan, L., & Shum, H.-Y. (2007, July). Image deblurring with blurred/noisy image pairs. *ACM Transactions on Graphics*, 26(3). doi:10.1145/1276377.1276379