Applications of Fourier Analysis in Image Recovery

Kang Guo TJHSST Computer Systems Lab 2009-2010

Introduction

Motion blur in images is a common problem for professionals in various fields. When the image is deblurred, the usefulness of the image increases. Parts of the image that were difficult to identify can be rendered to effective clarity. This project will explore and implement image deblurring techniques. By implementing these techniques, users can efficiently remove blur from an image.

Implementation

A Discrete Fourier Transform (DFT) is often too slow to be of practical use. Speed can be improved by implementing a Fast Fourier Transform (FFT). Instead of the N 2-dimensional transforms that a DFT uses, a FFT only performs 2N 1-dimensional transforms. Speed improves on the order of NlogN compared to N^2 for a DFT.

A Fourier Transform will transform an image in the spatial domain to the frequency domain, allowing for easier manipulation and a better view of key components in the image. By taking Fourier Transforms of a clear image and a blur filter, we can create a blurry image. The Fourier Transform of the blur filter, also known as a Point Spread Function (PSF), can be multiplied to the Fourier Transform of the original image. The actual values that are multiplied are complex and then must be inverse Fourier Transformed so that the final blurry image can be displayed. Many different types of blur can be modeled with a PSF. Therefore, if the PSF of a blurry image is known, then the image can be rather easily deblurred.

If multiplying the PSF and the Fourier Transform of the image creates a blurry image, then dividing the Fourier Transform of the blurry image and the PSF will reverse the convolution process. Knowing this, a blurry image can easily be deblurred with a non-blind deconvolution process. This process should be considered relatively simple, as the PSF is known. In cases where the PSF is not known, blind deconvolution processes must be applied, where the PSF must be estimated iteratively. Even non-blind deconvolution is not perfect and the deblurred image will contain noise that is created from the division of small Fourier values.

Problems with non-blind deconvolution

When a Fourier Transform is applied on an image, displaying the magnitude of the Fourier values can be an issue, due to their scale. Some values may be too small or too large and can not be displayed properly as 8-bit pixel values. Applying a logarithmic transform can fix this problem. However, the existence of these small Fourier values becomes a problem again in the inverse process, in which the blurred image is divided by the PSF. Dividing by these small values amplifies noise, and in some cases, may cause unacceptable results. I find this to occur when image are blurred with Gaussian blurs and blurs that are small in size.

Regularization

The process of regularization involves introducing additional information into a problem to allow a proper solution. This is particularly useful in machine learning and inverse problems. In the case of image deblurring, the inverse part of the process, where we divide the blurred image by the Point Spread Function produces a loss of data. Regularization will usually exist as some sort of restriction, forcing the final solution to fall in a set boundary of answers.

Fig. 1 Original Image

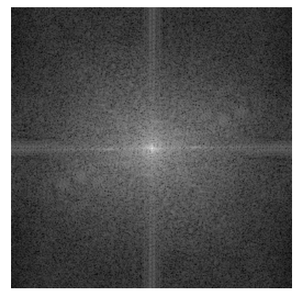

Fig. 2 Fourier Transformed Image

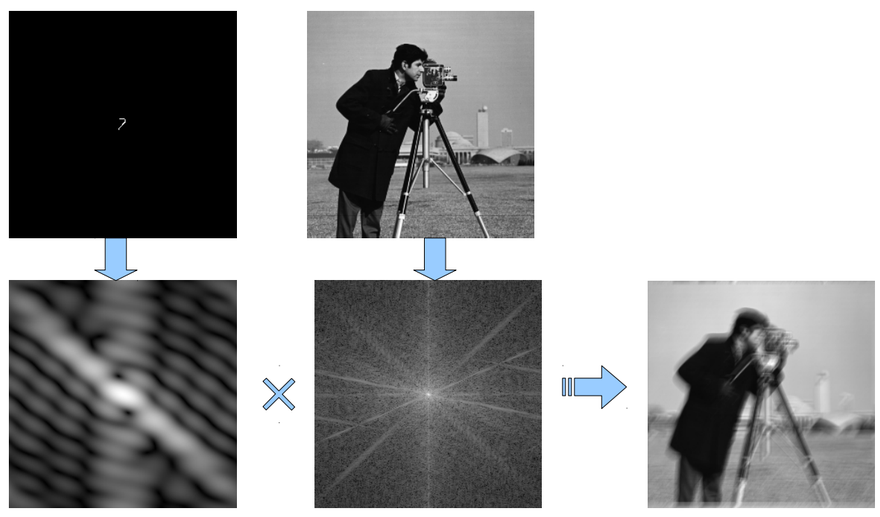

Fig. 3: Blurring an Image

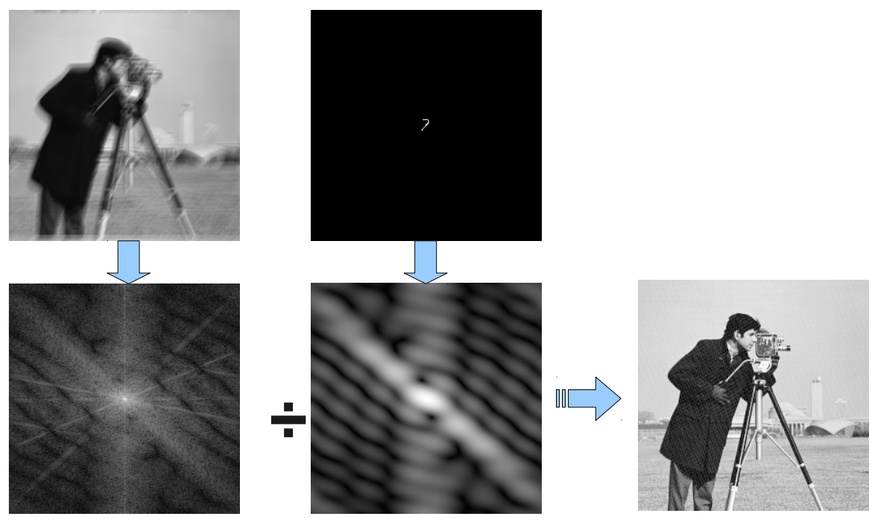

Fig. 4: Deblurring an Image

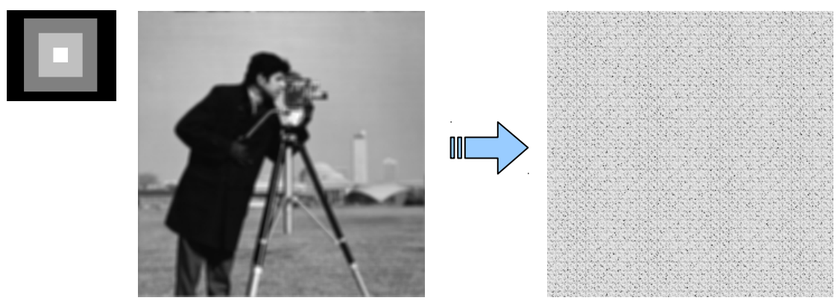

Fig. 5: Problems with Gaussian-like blurs in non-blind deconvolution

Matlab Implementation

Matlab is a good tool for general work with mathematics, but it also has an Image Processing Toolbox that is very helpful for image deblurring. Many functions necessary in the code that I have already written are implemented in Matlab. Due to the complexity of the deconvolution techniques presented thus far and the scope of this project, I implemented regularization in Matlab. Both of the Matlab functions *deconvreg()* (Fig. 8) and *deconvlucy()* (Fig. 7) implement a type of non-blind deconvolution. *deconvreg()* applies regularization to a normal non-blind deconvolution algorithm, while *deconvlucy()* estimates and update latent deblurred images iteratively.

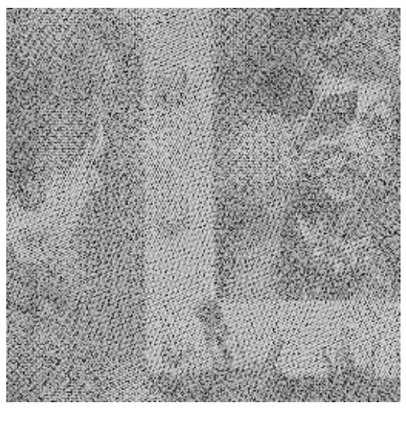

Fig. 6: No Regularization

Fig. 7: Richardson-Lucy

Fig. 8: Regularization