Assignment 4 - Filtering

CS 474

Submitted by Timothy Sweet on 12/2/13

Abstract:

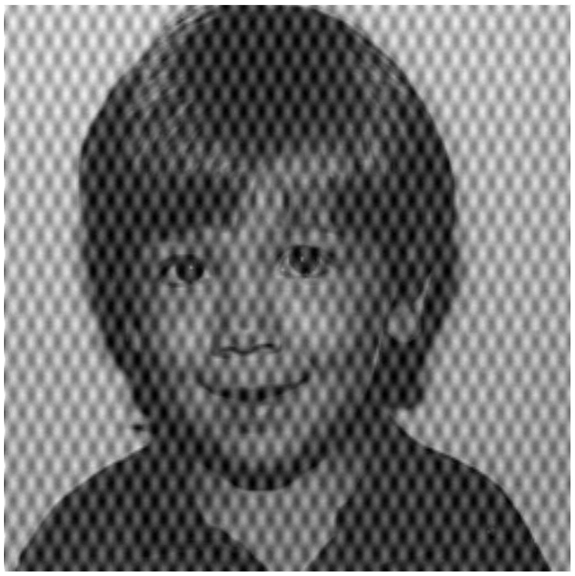
This report provides a summary of the results of three different image filtering algorithms. The first section demonstrates noise removal, where an image with periodic noise is restored to nearly its original version. The second section demonstrates convolution in the frequency domain, by applying a sobel mask to an image. The third section demonstrates image restoration under motion blurring.

Section 1 - Noise Removal

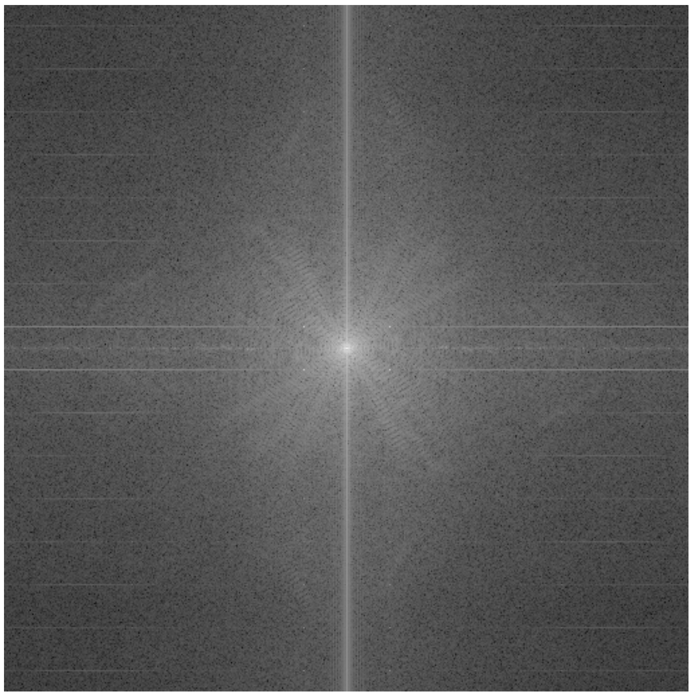
This section describes the results of performing noise removal on an image in the frequency domain. Specifically, the program implemented removes periodic noise. The program, in general, carries out these steps:

1. Retrieves the image and performs setup operations
2. Pads the image to an optimal size for the fourier transform
3. Computes the fourier transform of the image
4. Computes the magnitude of the fourier transform
5. Removes outlying points in the magnitude image: points which are significantly brighter than their four surrounding pixels.
6. Recreates the original image via the inverse fourier transform method.

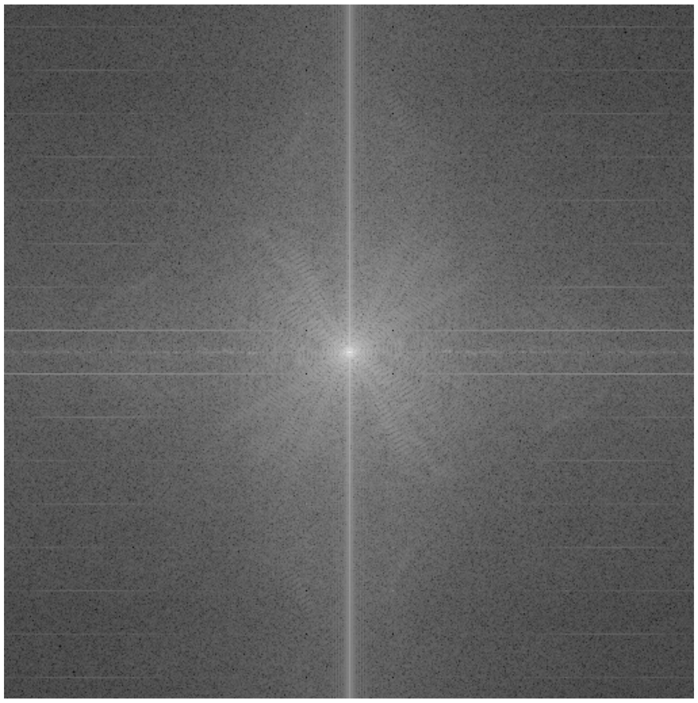
The following image is used as an input. This image is known to be corrupted with periodic noise:



The magnitude of the image above (stretch and normalized):

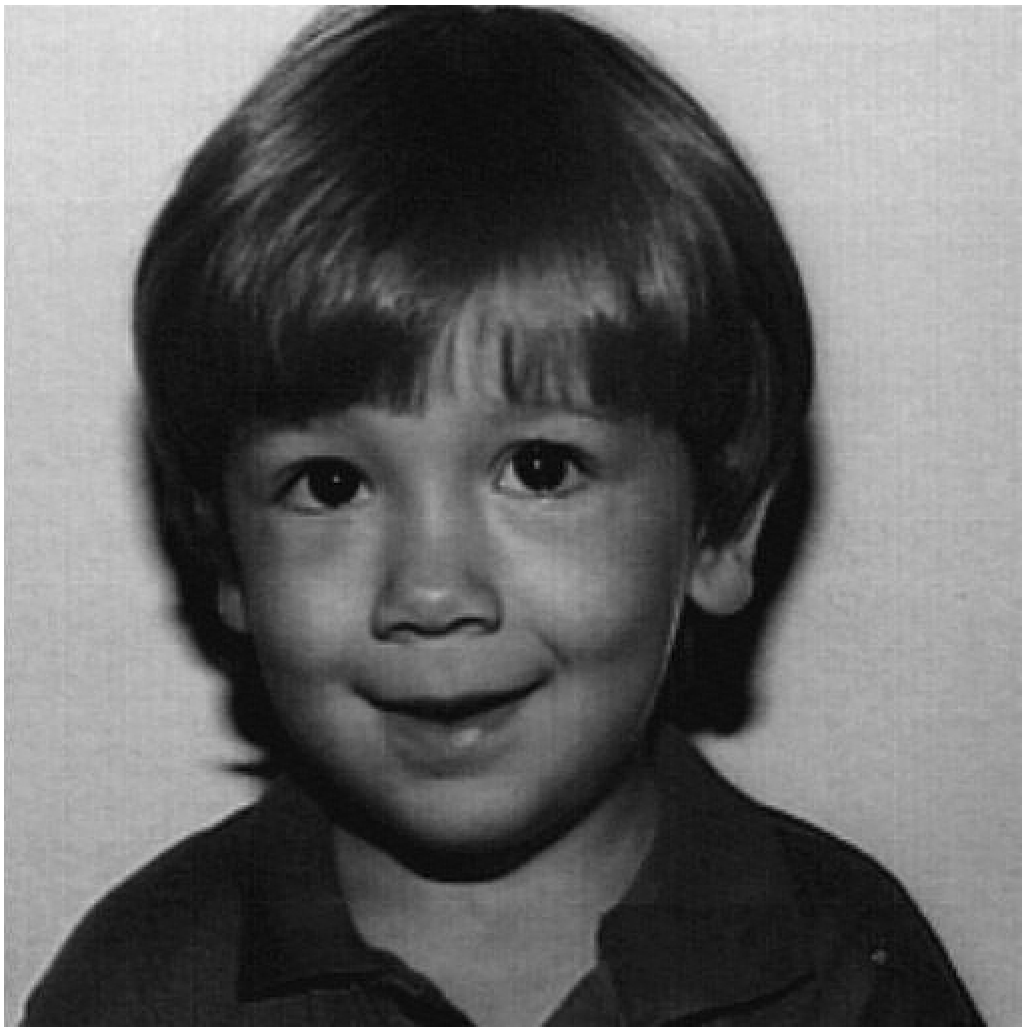


Notice there are four very bright pixels just outside the center of the image[1]
They are removed via the method described in step 5 above, to produce the following magnitude image:



[1] in the print version of this document they will not be visible, view the full definition version at: http://goo.gl/VE3ks4

Which produces the following output when transformed back into the spatial domain:

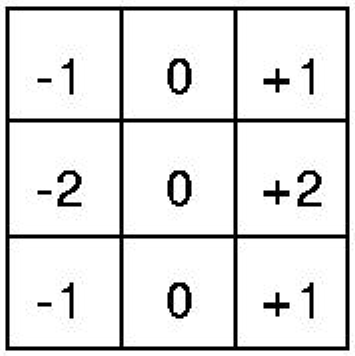


Notice the noise from the image is gone, but the information appears to be largely unchanged.

When the program is run on this input it actually removes 10 pixels from the magnitude image, including the four which are known to cause the noise. This does affect the output, but clearly not significantly as the image appears to be reconstructed very well. This could be avoided by manually selecting pixels to remove, but the automatic method is prefered for computational purposes.

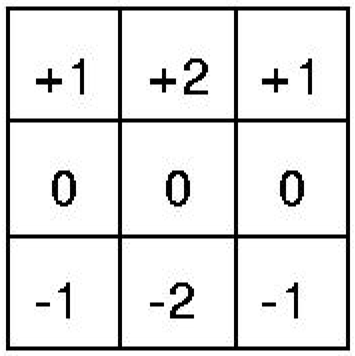
Section 2 - Convolution in the Frequency Domain

This section describes the results of applying the Sobel edge detector to an image in the fourier domain. The sobel edge detector is shaped as:



-1	0	+1
-2	0	+2
-1	0	+1

Gx



+1	+2	+1
0	0	0
-1	-2	-1

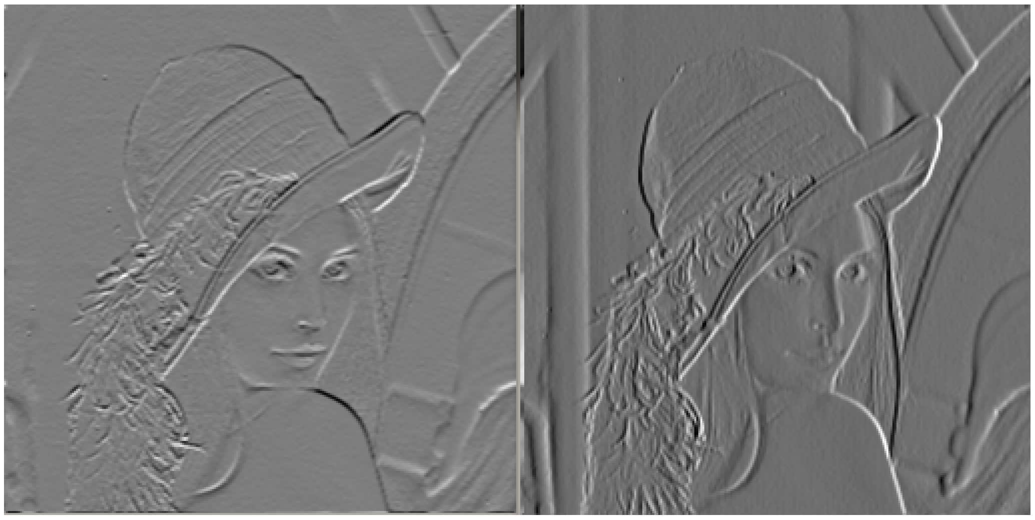
Gy

The program, in general, performs these steps:

1. Retrieves the image and performs setup operations
2. Pads the image to an optimal size for the fourier transform
3. Computes the fourier transform of the image
4. Sets up a complex image representing the Sobel edge detector, with the real part set to 0 and the imaginary part zero except in the center where the Sobel mask is placed.
5. Multiplies the image and mask in the fourier domain, which is equivalent to convolution in the spatial domain.
6. Repeats 4 and 5 for the transposed sobel mask.
7. Computes the gradient of the two results.
8. Recreates the original image via the inverse fourier transform method.
9. Normalizes the result to the displayable range.

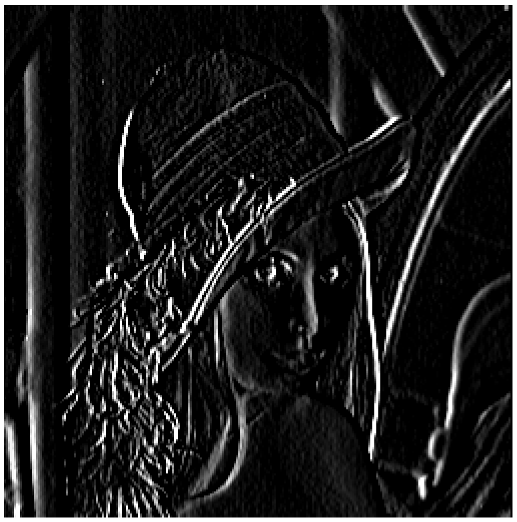
The input image used is:

Which is then processed as described in step 5 above to achieve the horizontal and vertical derivatives (these are visualized here for this report, but the step of transforming these back into the spatial domain is not necessary to achieve the end result). The first image is the vertical derivative, the second is the horizontal derivative:



And the magnitude of these images is computed and normalized to form:

For comparison, the sobel edge detector performed in the spatial domain is provided. It is observed that the images both express the strengths/weaknesses of the Sobel edge detector, most importantly the strength of being relatively resistant to noise, as opposed to another method such as the Prewitt edge detector.



Section 3: Image Restoration Under Motion Blur

This section describes the implementation of the inverse and Weiner filters in the fourier domain. Both filters are designed to restored images which are degraded due to linear time-invariant noise, such as motion blur.

The motion blur is modeled as:

$$H(u,v) = \frac{T}{\pi(u\alpha + vb)} \sin(\pi(ua + vb)) e^{-j\pi(ua+vb)}$$

Where H is the degradation model, T is 1, a, b are 0.1. This is simply the function:

$$g(x, y) = \int_0^T f[x - x_0(t), y - y_0(t)]\, dt$$

Where g is the distorted image, and f is the original image. The image is seen to be simply shifted through time.

The inversion filter performs this operation:

$$\hat{F}(u,v) = (1/H(u,v))G(u,v)$$

Where $\hat{F}$ is an approximation of the original image, H is the degradation model, and G is the original image under degradation. This attempts to directly remove the motion, but is not robust to noise. A better method is the Weiner Filter:

$$\hat{F}(u,v) = \frac{|H(u,v)|^2}{|H(u,v)|^2 + K} \cdot \frac{G(u,v)}{H(u,v)}$$

Where G, H are the same as above, and K is some constant. This attempts to model the image motion similar to a least mean squares filter.

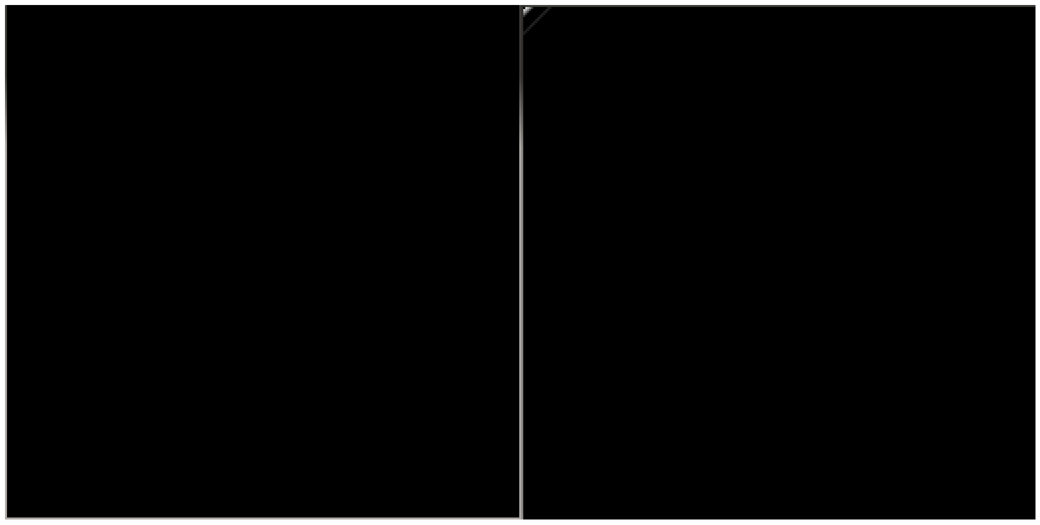
The program, in general, performs these steps:

1. Pads the image to an optimal size for the fourier transform
2. Computes the fourier transform of the image
3. Generates H, the motion blur model.
4. Distorts the original image with H
5. Adds a small amount of noise to the distorted image
6. Attempts to recover the original image by inversion filtering.
7. Attempts to recover the original image by Weiner filtering.
8. Recreates the original image via the inverse fourier transform method for both filtering methods.

9. Normalizes the results to the displayable range.

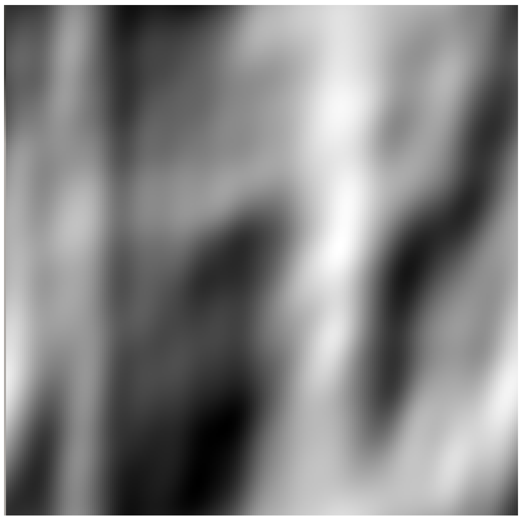
The input image used is:

The motion blur model produced by step 3 above, where the left side is the real part (all zero) and the right part is the imaginary part[2]:



[2] In case the images are difficult to see on paper: the right side image has a small diagonal line across the top left corner. See the full resolution document as mentioned earlier for details.

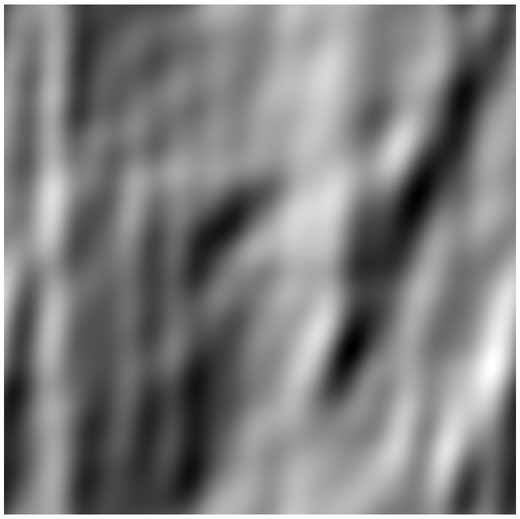
Which produces the following blurred image:



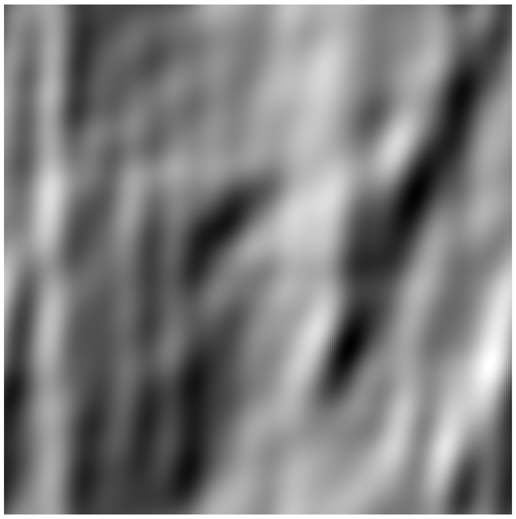
It is noted that this motion seems quite extreme, however the results have been checked multiple times both manually and using Matlab, and are verified to conform to the instructions precisely. There is a small complexity (which doesn't seem to affect the results)
in which the upper left corner of the motion model cannot be computed because it requires a division by zero. The program sets this value to zero.

As a side note, the following images are produced under vertical and horizontal blur, respectively:

Inversion filtering on the bidirectional blured image produces:



And Weiner filtering with the same distorion and K=1 produces:



It is observed that there is very little information recovered from these images, due to the fact that there is such extreme motion blur in the input image.

If the blur parameters a and b are lowered to 0.005 the results are greatly improved. The left image is the blurred image, the center is the image filtered with inversion filtering, and the third image is filtered with Weiner filtering.