CS3IA16 coursework assignment 1 – Image Enhancement

Abstract

This report aims to detail the steps that were taken in order to implement image enhancement in the spatial and frequency domains as well as the results that were obtained by using different enhancement techniques. The image enhancement is performed on a gray level image that has been distorted by a combination of periodic and random noise. The original gray level image is provided to assist in the evaluation of the results. To evaluate the results a comparison is made between the restored images that are created with the application of algorithms on the distorted image and the original image by using the measure of Average Mean Square Error (AMSE). In the end a sequence of different image enhancement techniques will be discovered which will be applied to the distorted image and result in a restored image that is close to the original one.

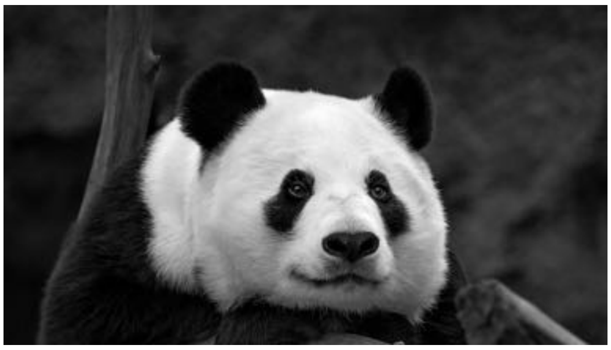
Introduction

To begin, the programming environment that was chosen for this project was MatLab since it provided numerous different techniques that could be applied to the distorted image in order to enhance it. The first step was to load the original gray level image and the image with the noise to the program as

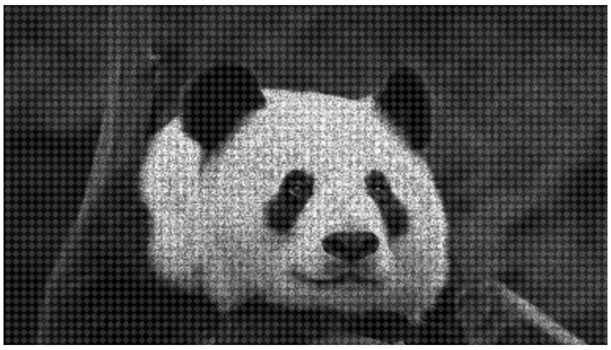
```
I = imread('PandaNoise.bmp');
I0 = imread('PandaOriginal.bmp');
```

and to make sure they could be accessed by the command `imshow`, which was successful.

PandaOriginal



PandaNoise



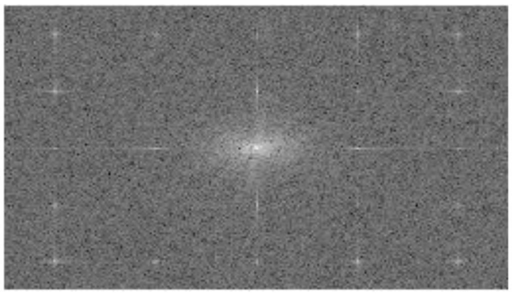
In the images provided above, PandaOriginal highlights the original image of the Panda, whereas PandaNoise details this same image in which noise had been added. The purpose of the image enhancement techniques mentioned throughout this report will be to aim to move as much of the noise as possible from the image displayed in PandaNoise in order to achieve a close match to the original image. In order to assess the quality of the image after the application of the image enhancement techniques, the metric of Average Mean Square Error will be utilised.

Methodology

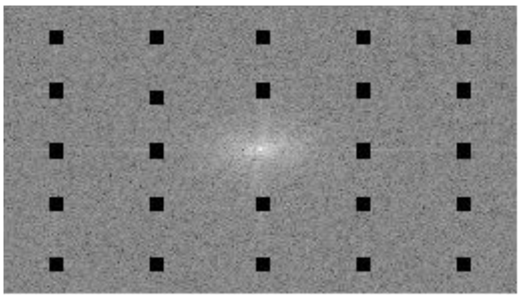
Removal of Periodic Noise and Random Noise

The initial step in order to remove the periodic noise was the implementation of a Fast Fourier Transform (FFT) which takes in an image in the spatial domain and translates it into the frequency domain. This is an efficient version of the Discrete Fourier Transform (DFT). In MatLab this can be implemented with the function fft2. After carrying out the FFT on the image, another function fftshift is required in order to shift the zero (0) frequency component to the centre. A logarithmic function is then applied to the output in order to visualise the magnitude spectrum.

Magnitude Spectrum



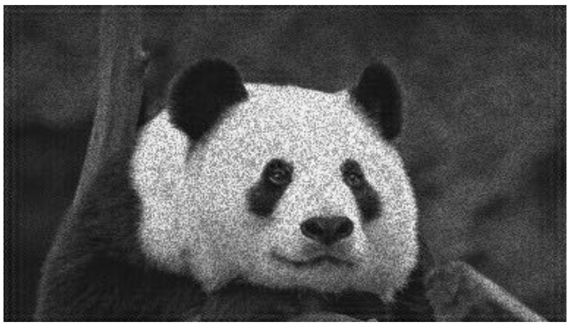
Band-stop Filter



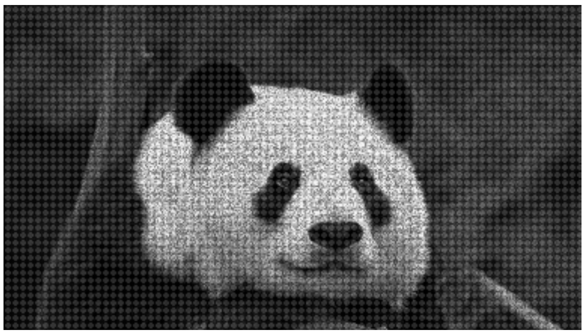
The image above on the left details the results of applying the FFT to the image with noise. As there is periodic noise in the image, this is represented in the Fourier spectrum via various peaks (highlighted white areas) in the image. In order to remove the periodic noise from the image, a filter can be applied across the image to reject the frequencies in which the periodic noise occurs. This can be visualised in the image above on the right which details the Band-stop filter implemented in the frequency domain. This was implemented by altering the values of the array which holds the image of the magnitude spectrum and setting the value to 0, where a peak appeared on the image.

Having implemented a filter in the frequency domain, an inverse FFT must now be applied to the image with the filter over-layed to return the image to the spatial domain. This can be achieved through utilising the MatLab function ifft2. After implementing the inverse FFT, the resulting image can be seen below.

Magnitude Spectrum Enhanced Image



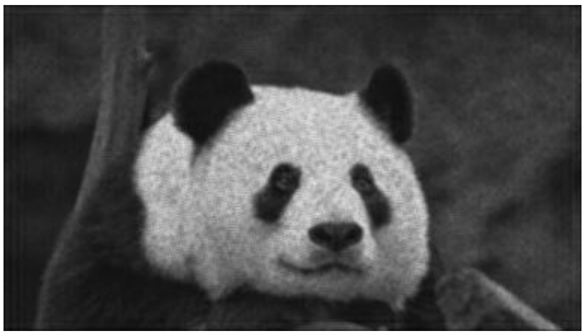
PandaNoise



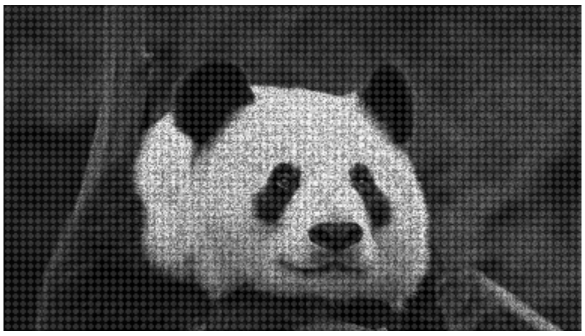
It is clear that after the FFT is applied and the filter in the frequency domain, the resulting image appears to have significantly less noise than the noise image that was given.

After converting the image back into the spatial domain, other filters can then be applied in order to reduce the random noise in the image and work toward improving the image quality. The filter used on the enhanced image is an averaging filter. This works by convolving a window across the image which calculates the mean average of the pixels relative to their neighbours. This helps to reduce noise as any pixels which are very different from their neighbours will have their intensity values decreased.

Averaging Filter



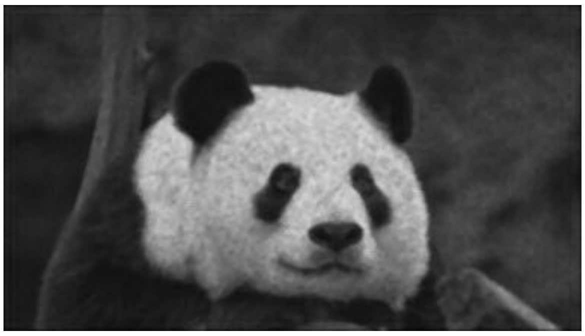
PandaNoise



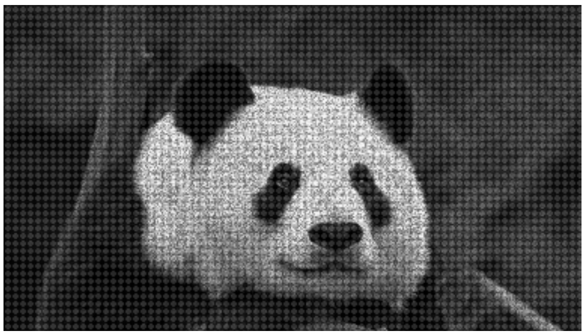
After the application of the averaging filter, there is a great reduction of noise in the image.

Following applying the averaging filter, a median filter was applied in the spatial domain. Median filtering is a non linear method used to remove noise from images. This is particularly effective in removing noise such as salt and pepper noise and removing noise while preserving edges. The median filter works by convolution. The filter convolves through the image pixel by pixel and replaces each value with the median value of neighbouring pixels. This is calculated by first sorting all of the pixel values from the window in numerical order and then replacing the pixel being considered with the middle (median) value. Having implemented this, the results can be seen below.

Median Filter



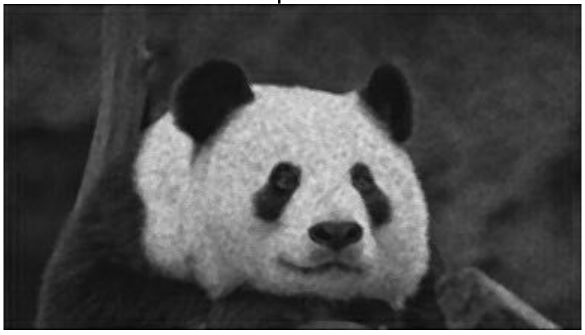
PandaNoise



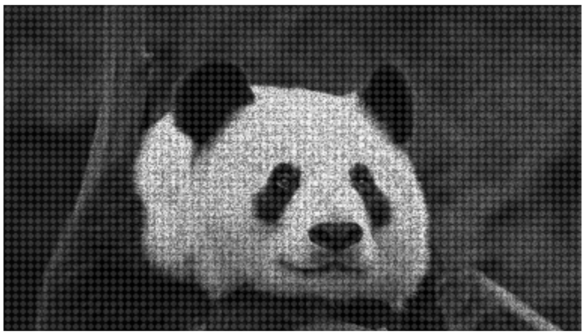
The median filter smoothes the enhanced image even more by reducing further noise from the image, therefore increasing the image quality.

The last image enhancement technique that was applied was sharpening the image. This is handled by the MatLab function `imsharpen`. This works by sharpening the grayscale input image by using the unsharp masking method. This method comes from a publishing industry process in which an image is sharpened by subtracting a blurred (unsharp) version of the image from itself.

Sharpened



PandaNoise

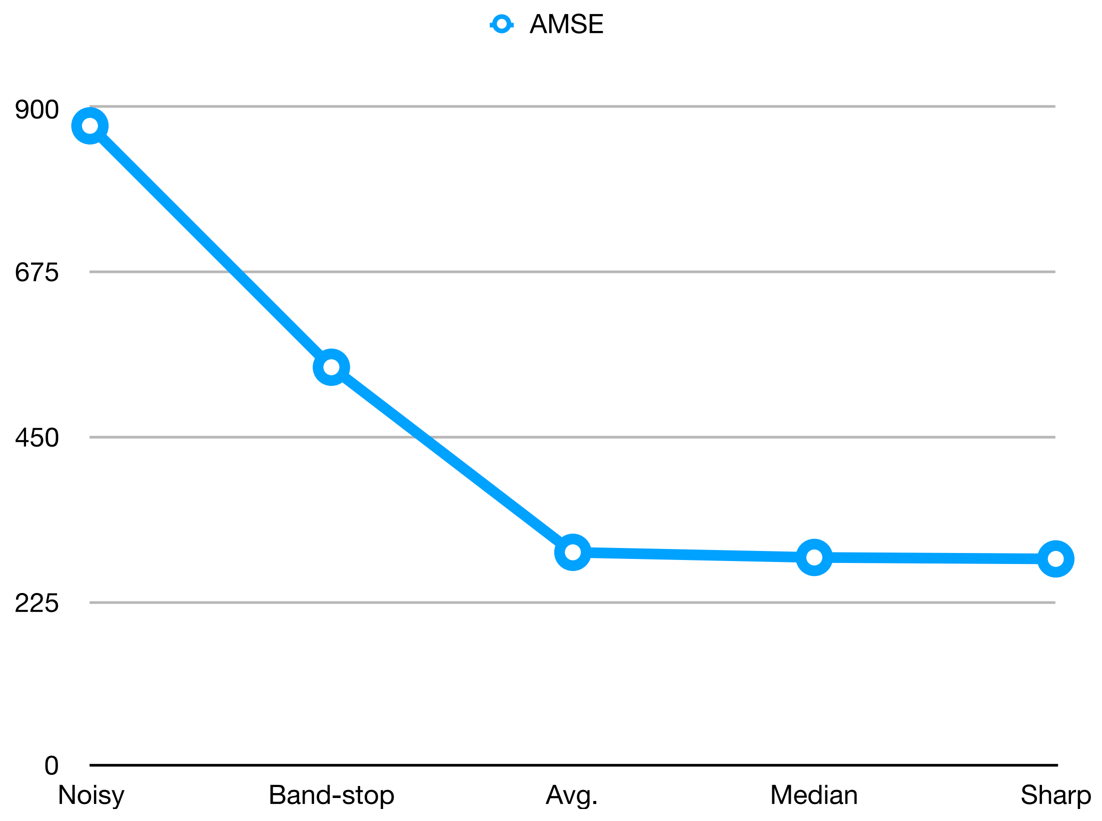


Results and Discussion

The results that were obtained were satisfactory since the Average Mean Square Error that was initially 875.2103 from the comparison between the original and the noisy image, dropped to 283.2705 after the application of all the image enhancement techniques.

Firstly, the comparison between the noise image and the image resulting from the FFT and the magnitude spectrum gave an AMSE of 545.37786 which was a significant improvement. Following the application of the averaging filter to the enhanced image the AMSE from the comparison to the noise image decreased drastically to 292.2824. After applying the median filter to the new enhanced image the AMSE continued to decrease reaching 285.5434, although it was not as dramatic a difference as on the previous comparisons. Lastly, by sharpening the enhanced image and comparing it to the noise image an AMSE of 283.2705 was achieved which was the optimum result found so far. Other image enhancement techniques were considered as well. For example, instead of using an averaging filter after applying the band-stop filter, a gaussian filter was utilised. However unfortunately this did not provide results as strong as the ones given from the averaging filter, as the gaussian filter gave an AMSE of 299.8908 in comparison to the 292.2824 from the averaging filter. Furthermore, the image was also sharpened a second time after the initial image sharpening in an attempt to further reduce the noise in the image and increase the image quality. This technique did also not improve the result and provided an AMSE of 365.7835 instead.

Overall, the best method found for reducing the noise and improving the quality of the image by applying filters in both the spatial and frequency domains was to firstly apply a band-stop filter in the frequency domain. Having converted the image back to the spatial domain with the inverse FFT, an averaging filter was applied, followed by a median filter and finally by a sharpening image enhancement technique. The results of the reduction in noise and therefore a decrease in the AMSE shown through the process of image enhancement can be visualised in the graph below.



Conclusion

In conclusion, there are various different image enhancement techniques that can be used to reduce periodic and random noise across the frequency and spatial domains. The final result of the enhancement can differ according to the techniques that are utilised as well as the order that they are used in. It could be possible that there is a more optimal order in order to use the techniques implemented, however through trial and error and analysing the AMSE at each step, the order of the techniques implemented was found to be the most efficient.

The techniques implemented could have been further improved upon, such as in the case of implementing the band-stop filter in the frequency domain. The values given for the filter were developed given a visual implementation of the image, however it could be possible to construct an algorithm which analyses Fourier spectra and the respective peaks of the image and then constructs a band-stop filter depending on the organisational structure of the image. If this sort of algorithm could have been implemented as well, the resulting filter could be much more successful and therefore then provide a much stronger result with a lower AMSE.

All in all, the requirements of the specification were met since the algorithms that were used improved the quality of the distorted image by removing a significant amount of the noise and the results that were evaluated by comparing the restored images with the original image managed to drastically decrease the Average Mean Square Error and therefore allowed the final restored image to get as close to the original image as possible.

APPENDIX

```
I = imread('PandaNoise.bmp');
I0 = imread('PandaOriginal.bmp');
I2 = imread('MagnitudeSpecMask.bmp');
I3 = imread('AveragingFilter.bmp');
I4 = imread('MedianFilter.bmp');
I5 = imread('Sharpened.bmp');
I6 = imread('gaussianinsteadavg.bmp');
I7 = imread('Sharpened2.bmp');

gray_img1 = rgb2gray(I2);
gray_img2 = rgb2gray(I3);
gray_img3 = rgb2gray(I4);
gray_img4 = rgb2gray(I5);
gray_img5 = rgb2gray(I6);
gray_img6 = rgb2gray(I7);

%Create FFT
spec_orig = fft2(double(I));

%Shift the zero frequency component to the centre
spec_img = fftshift(spec_orig);

%Create band-stop filter(cover white peaks in the magnitude spectrum to
remove periodic noise)
for j = 110:120
    for n = 20:30
        spec_img(n,j) = 0;
    end
    for n = 65:75
        spec_img(n,j) = 0;
    end
    for n = 105:115
        spec_img(n,j) = 0;
    end
    for n = 145:155
        spec_img(n,j) = 0;
    end
    for n = 190:200
        spec_img(n,j) = 0;
    end

end
for j = 35:45
    for n = 20:30
        spec_img(n,j) = 0;
    end
     for n = 60:70
        spec_img(n,j) = 0;
    end
    for n = 105:115
        spec_img(n,j) = 0;
    end
    for n = 145:155
        spec_img(n,j) = 0;
    end
    for n = 190:200
        spec_img(n,j) = 0;
    end
end
for j = 265:275
     for n = 20:30
        spec_img(n,j) = 0;
    end
     for n = 60:70
        spec_img(n,j) = 0;
    end
    for n = 105:115
        spec_img(n,j) = 0;
    end
    for n = 145:155
        spec_img(n,j) = 0;
    end
    for n = 190:200
        spec_img(n,j) = 0;
    end
end
for j = 340:350
    for n = 20:30
        spec_img(n,j) = 0;
    end
     for n = 60:70
        spec_img(n,j) = 0;
    end
    for n = 105:115
        spec_img(n,j) = 0;
    end
    for n = 145:155
        spec_img(n,j) = 0;
    end
    for n = 190:200
        spec_img(n,j) = 0;
    end
end
for j = 190:200
    for n = 20:30
        spec_img(n,j) = 0;
    end
     for n = 60:70
        spec_img(n,j) = 0;
    end

    for n = 145:155
        spec_img(n,j) = 0;
    end
    for n = 190:200
        spec_img(n,j) = 0;
    end
end

%Getting Back the Image
%figure;subplot(2,1,1);
%spec_img = log(1 + spec_img);
%imshow(spec_img,[]);
```

```matlab
%Create inverse FFT
ptnfx = real(ifft2(ifftshift(spec_img)));
%figure;
%axes('Units', 'normalized', 'Position', [0 0 1 1]);
%imshow(ptnfx, []);

%Averaging filter
Kaverage = filter2(fspecial('average', 3), ptnfx)/255;
%figure;
%axes('Units', 'normalized', 'Position', [0 0 1 1]);
%imshow(Kaverage)

%Median filter
Kmedian = medfilt2(Kaverage);
%figure;
%axes('Units', 'normalized', 'Position', [0 0 1 1]);
%imshow(Kmedian);

%Sharpening
Sharp = imsharpen(Kmedian);
figure;
axes('Units', 'normalized', 'Position', [0 0 1 1]);
imshow(Sharp2);

%%MSE calculation by comparison of the images
err = immse(gray_img4, I0);
fprintf('\n The Mean Square Error is %0.4f\n',err);
```