**Fundamentals of Image Processing with MATLAB – Assignment 1**

Task 1: Nonlinear image filtering

The first task intended to perform the diffusion filter through the non-linear iterative process a series of image files. This entailed sharpening the edges and removing the little details within the grey scale images. This is demonstrated through the code in Figure 1 that signifies the equation for the iterative process.

Text

Description automatically generated

Figure 1 Image Filtering Iteration

The iterations operate through the is apparent through two pictures in Figure 1. Edges are discovered based on the difference between the inner brighter regions from the image compared to their specific darker outer region. These symbolise gradients that are identified by locating white pixels in the centre that are surrounded by black pixels. However, the sharpening resulted in a blurring effect for the neighbouring pixels with a closer colour, represented as a value, to the central pixel.

The outputs are generated through the executed program that outputs each image presented as figures from the procedure. This enabled the parameters for the constant (K) and iterations (N) to change and affect the imported image.

First, the iterations begins with N = 10 and introduces the constant K = 50. This illustrates the minimum number of iterations that results in a little number of changes made to the image.

Graphical user interface, website

Description automatically generated

Figure 2 Trui.tif with N = 10 || K = 50

An increased number of iterations for N = 100, performed more image filtering operations with the constant as the K = 50. This results in the figure having sharper edges and creating a blurrier effect.

Graphical user interface

Description automatically generated

Figure 3 Trui.tif with N = 100 || K = 50

A lower constant value, with K = 5, and the same number of iterations N = 10 causes the program struggle to identify the edges within the imported image in Figure 4.

Graphical user interface

Description automatically generated

Figure 4 Trui.tif with N = 10 || K = 5

A higher constant K = 500, with the same N = 10, provides a sharper change in the greyscale value to identify the edges. Therefore, the image is not altered too much from the original.

Graphical user interface

Description automatically generated

Figure 5 trui.tif with N = 10 || K = 500

Here’s the image filtering performed on the remaining images…

Graphical user interface, application

Description automatically generated

Figure 6 pout.tif with N = 100 || K = 50

Graphical user interface

Description automatically generated

Figure 7 barbara.tif with N = 100 || K = 50

Graphical user interface

Description automatically generated

Figure 8 cameraman.tif with N = 100 || K = 50

A picture containing text, screenshot, suit, person

Description automatically generated

Figure 9 einstein.tif with N = 100 || K = 50

A picture containing text, appliance, white goods

Description automatically generated

Figure 10 newborn.tif with N = 100 || K = 50

Task 2: Low-light image enhancement

The second task continues upon the previous image filtering assignment to perform enhancements on low-light image. Through the same iterative process, the dominator as U (x, y) was found from obtaining T(x, y) to enhance an imported image. First the procedure illustrates discovering the max value in Figure 11 to repeat the iterative process and execute the equation to enhance the pixels.

Text

Description automatically generated

Figure 11 Task 2 Finding Max Iteration

After the iterative process, the enhancement equation was implemented to enhance the pixels within the imported image in Figure 12.

Graphical user interface, text, application

Description automatically generated

Figure 12 Enhancement Equation Iteration

When the program runs, the first figure presents the original image and outputs the second figure with the enhanced image. This generates this series of figures show here…

A picture containing text, indoor, screenshot

Description automatically generated

Figure 13 8.bmp Enhancement

Graphical user interface, application

Description automatically generated

Figure 14 10.bmp Enhancement

A picture containing text, indoor, monitor, television

Description automatically generated

Figure 15 30.bmp Enhancement

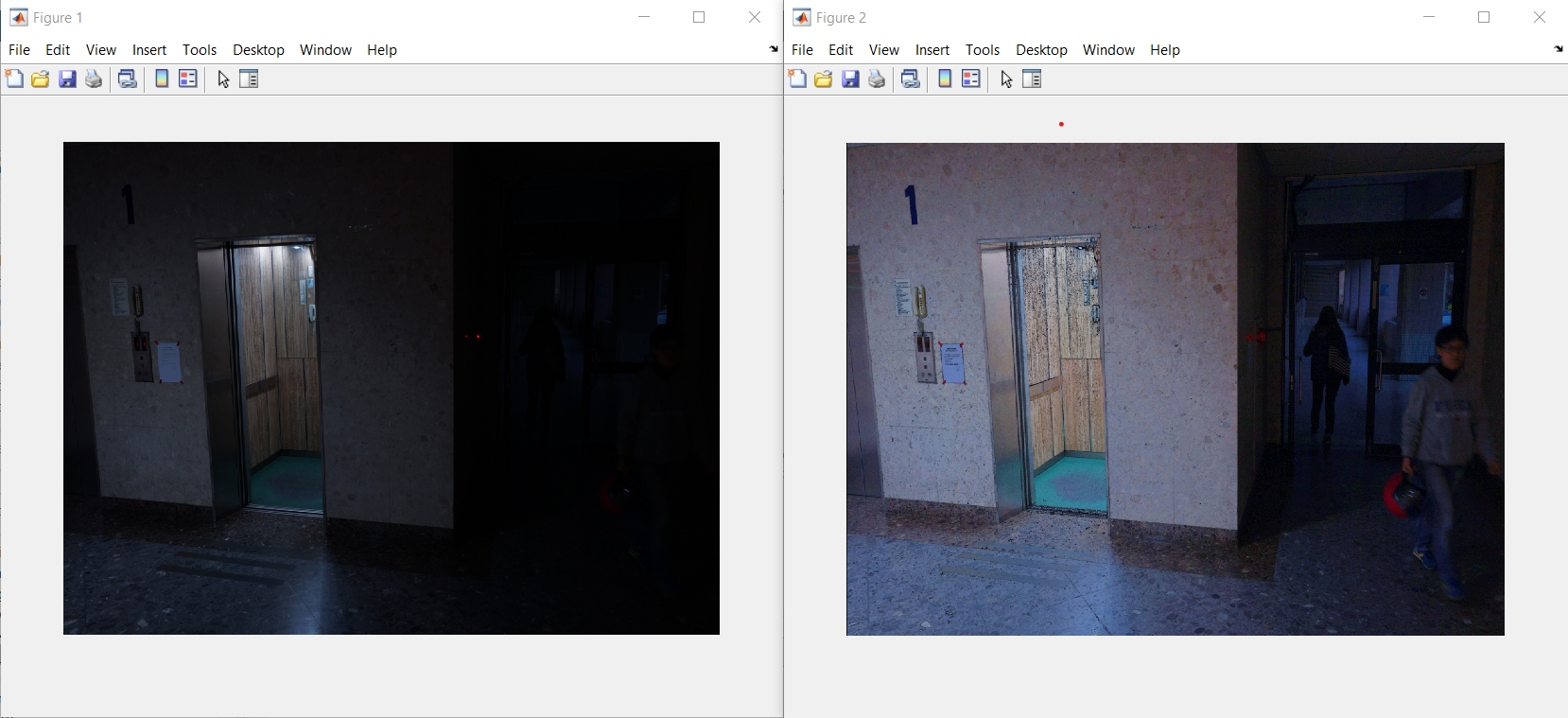


Figure 16 scene1.bmp Enhancement

Task 3: Image filtering in frequency domain.

The final tasks incorporate Fourier transforms on image filtering that enables the restoration for.bmp files corrupted by periodic noise. Through the Fourier4ip.m MATLAB script, the function ffshift intends to shift the zero-frequency component of an image to the centre of spectrum. From a spatial image, the Fourier transform can breakdown the image into their sine and cosine components to output the frequency domain image. These have four small crosses that correspond to the frequencies behind the periodic noise identifies the unwanted frequencies. Moreover, the band-stop filter should be constructed to remove the noise while preserving the image’s quality.

Initially, the imported image was corrupted had periodic noise and initialised as a Fourier image for the algorithm inputs to undergo the fftshift function. The iteration gradually goes through the size of the image to create the domain frequency plotted with the zero-frequency moved to the centre of the spectrum. Therefore, the user can choose and store certain (x, y) values into an array, where the corrupted image can be identified by clicking the bright spike on the frequency domain image.

The band-stop filter eliminates the frequency component for the predefined neighbouring centre frequency through the binary filter that sets the response either 0 or 1. Therefore, the spikes on the images are stored by the algorithm that iterates through the picture to store the position as x and y values. This entails a conditional statement for the formula that discovers the pixels within the predefined area. Figure 17 demonstrates the value is set to 0 to symbolise the removal for periodic noise.

Text

Description automatically generated

Figure 17 Periodic Noise Removal Iteration

Inverse Fourier Transform restores the image without the periodic noise in Figure 18

A picture containing text

Description automatically generated

Figure 18 Frequency domain with Restored Image

In conclusion, the procedure demonstrated the removed periodic noise to restore the image. Through the series of figures, the original image has the corresponding frequency domain with the restored image. For example, the frequency domain presents the period noise with the four frequencies, where the periodic noise is removed. This is highlighted through the diagonal lines are removed on the original

As the periodic noise are selected on the frequency domain, the restored image is shown on the subsequent figure. To compare the difference between, Figure 19 displays light switch with the diagonal lines from the restored image are less prominent in the original image. This suggests that periodic noise has been removed and the image was restored.

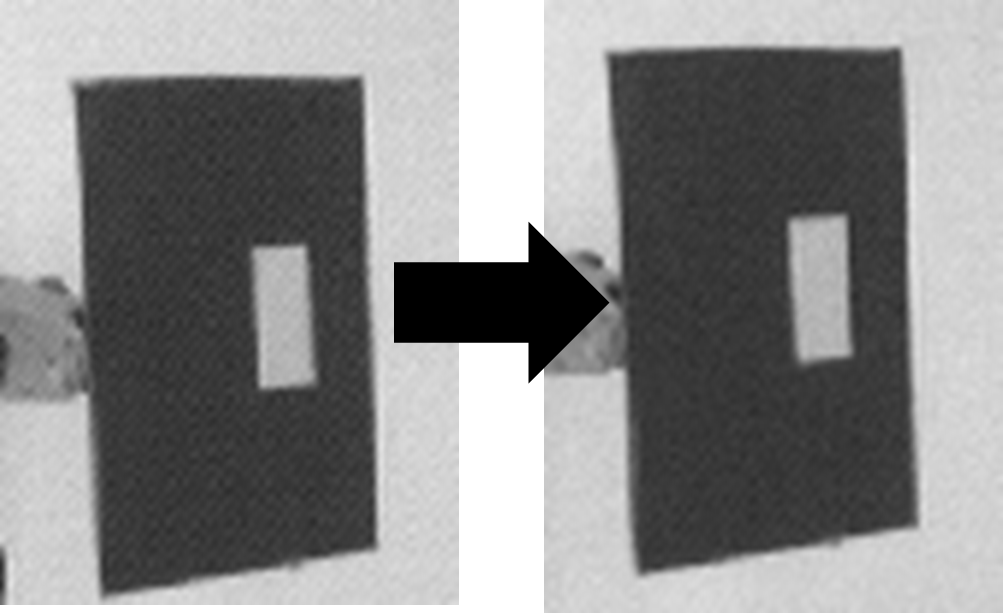


Figure 19 Manified Image Restoration