IMPERIAL COLLEGE of SCIENCE, TECHNOLOGY & MEDICINE

DEPARTMENT of ELECTRICAL & ELECTRONIC ENGINEERING

MSc in Communications and Signal Processing

Formal Report No. 2

Name	Stefanos Agamemnon Frilingos
Experiment Code	TS
Title of Experiment	Digital Image Processing Methods
Date of Submission	January 6, 2025
Supervisor of Experiment	Nikolaos Giakoumoglou
Grade	

Communications, Control and Signal Processing Laboratory

Contents

1 Introduction

The TS lab experiment from the Image Processing module aims at developing the intuition behind the use of image transforms, enhancement techniques, compression and restoration. An independent design exercise that merges a few of the key concepts explored in this report will be presented towards the end of the experimental process section. Since the experiment involves various methods of image processing, it is worth noting that all images used were either imported from the default MATLAB image stock archive, or designed by myself, as will be shown in Part I of the experimental process, involving artificial images.

2 Relevant Background Theory

The experiment is divided into five parts. This section provides a concise overview of the relevant theory necessary to understand each part, with an emphasis on brevity due to the extensive use of images in the report.

2.1 Part I: Image Transforms

This part compares the FFT, DCT, and Hadamard transforms, highlighting their fundamental properties and applications.

2.1.1 Fast Fourier Transform (FFT)

The FFT computes the 2D Discrete Fourier Transform (DFT), transforming an image from the spatial domain to the frequency domain. This separates high-frequency components (edges and details) from low-frequency components (smooth regions). The 2D DFT is expressed as:

$$X[u,v] = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f[x,y]e^{-j2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)}$$

where $f[x,y]$ is the intensity at spatial coordinates (x,y), $X[u,v]$ represents the frequency-domain coefficients, and M,N are the image dimensions.

The FFT significantly reduces computational complexity from $O(M^2N^2)$ to $O(MN\log(MN))$, making it essential for applications like edge enhancement, noise reduction, and pattern recognition.

2.1.2 Discrete Cosine Transform (DCT)

The DCT, unlike the FFT, uses only real numbers, making it computationally efficient and ideal for compression algorithms like JPEG. It concentrates most image energy into a few low-frequency components, defined as:

$$C[u,v] = \alpha(u)\alpha(v)\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f[x,y]\cos\left(\frac{\pi u(2x+1)}{2M}\right)\cos\left(\frac{\pi v(2y+1)}{2N}\right)$$

where $\alpha(u)$ and $\alpha(v)$ are normalization factors:

$$\alpha(u) = \begin{cases} \sqrt{\frac{1}{M}}, & \text{if } u = 0 \\ \sqrt{\frac{2}{M}}, & \text{if } u > 0 \end{cases}$$

The DCT's energy compaction capability enables significant storage reductions while maintaining perceptual quality.

2.1.3 Hadamard Transform (DHT)

The Hadamard Transform is a non-sinusoidal, orthogonal transform that uses binary basis functions. It is computationally efficient and well-suited for binary images or those with sharp transitions. The 2D Hadamard Transform is defined as:

$$H[u, v] = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f[x, y] W(x, u) W(y, v)$$

where $W(x, u) = (-1)^{\text{bitwise XOR}(x,u)}$. While simple and fast, its energy compaction is less effective than the DCT, limiting its use in compression tasks.

2.2 Part II: Image Enhancement

2.3 Histogram

Histograms play a vital role in image enhancement by representing intensity level frequencies. Mathematically, a histogram $H[i]$ is defined as:

$$H[i] = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \delta(f[x, y] - i), \quad i = 0, 1, \ldots, L-1$$

where $\delta(f[x, y] - i)$ equals 1 when $f[x, y] = i$, otherwise 0. Techniques like histogram equalization redistribute intensity values for improved contrast, with applications in medical imaging, remote sensing, and computer vision.

2.3.1 Edge Detection Methods

Sobel Method The Sobel operator calculates image gradients using convolution masks, emphasizing edges with combined smoothing and differentiation.

Roberts Method This gradient-based method uses small 2×2 kernels, detecting edges by intensity differences between adjacent pixels. It is efficient but noise-sensitive.

Prewitt Method Similar to Sobel, the Prewitt operator uses constant-weight kernels for edge detection in noiseless images.

LoG Method The Laplacian of Gaussian (LoG) applies Gaussian smoothing followed by the Laplacian operator to detect rapid intensity changes, identifying edges as zero-crossings.

2.3.2 Median Filtering

Median filtering reduces noise by replacing pixel intensities with the median of neighboring pixels. It excels at removing salt-and-pepper noise while preserving edges, but larger blocks risk blurring details.

2.4 Part III-IV: Image Compression and Design Experience

The DCT is employed for compressing images using varying thresholds and block sizes. Smaller blocks preserve details but require lower thresholds, while larger blocks allow higher thresholds with some quality loss.

2.5 Part V: Image Restoration

2.5.1 Inverse Filtering

Inverse filtering restores degraded images by applying the inverse degradation function:

$$F(u,v) = \frac{G(u,v)}{H(u,v)}, \quad H(u,v) \neq 0$$

Here, $H(u,v)$ is the degradation function, and $F(u,v)$ is the Fourier transform of the image.

2.5.2 Wiener Filtering

Wiener filtering minimizes mean squared error by incorporating noise and signal power spectra:

$$\hat{F}(u,v) = \frac{S_{ff}(u,v)H^*(u,v)}{S_{ff}(u,v)|H(u,v)|^2 + S_{nn}(u,v)}Y(u,v)$$

where $S_{ff}(u,v) = |F(u,v)|^2$ is the power spectral density of the original signal, $S_{nn}(u,v) = |N(u,v)|^2$ is the power spectral density of the noise, $H(u,v)$ is the degradation function, and $Y(u,v)$ is the degraded image in the frequency domain.

It excels in restoring images degraded by both noise and blur, with applications in medical imaging and video processing.

3 Experimental Process

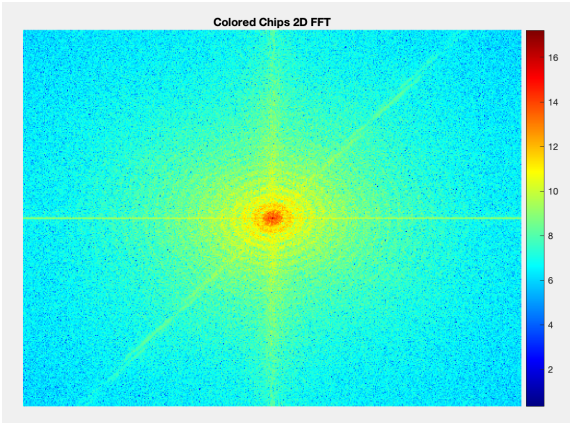
3.1 Part I: Image Transforms

The following section describes the experimental procedure with respect to the FFT, DCT and DHT.

3.1.1 Fast Fourier Transform (FFT)

a) The first task of part I sought to evaluate the frequency representation of an image. In this case, the log magnitude of the FFT for the colored chips MATLAB image was observed, as shown in Figure 1 below.

(a) Colored Chips MATLAB Image

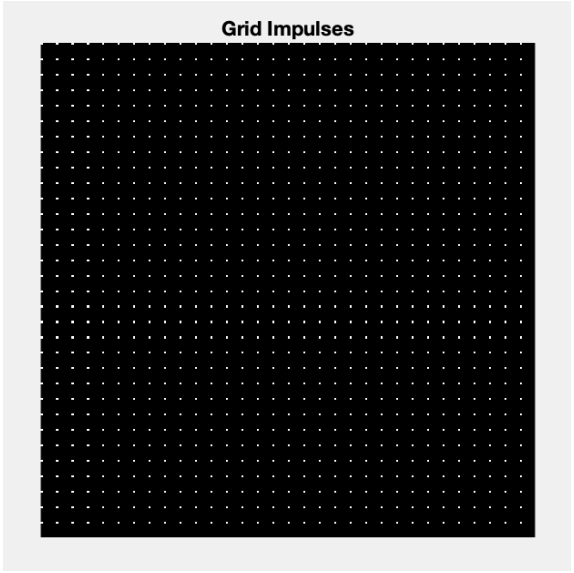


(b) Colored Chips 2D FFT

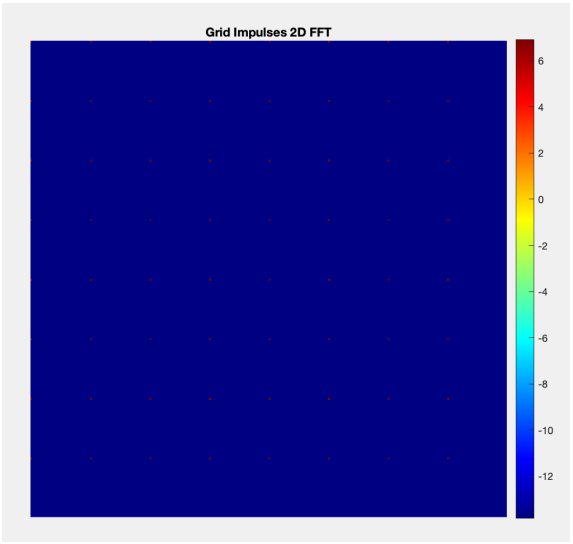
Figure 1: Colored Chips transformed with 2D FFT

It is observed how the majority of energy in the image is concentrated near the origin (the center of the 2D frequency representation) which refers to the lower index FFT coefficients. This occurs because lower frequencies dominate in most images without extensive noise. Low frequencies can be related to smooth regions while high frequencies are more common when there are lots of edges or rapid changes in an image, such as details or contrasting shapes. This aligns with the chips image 1a which contains mostly smooth colored regions.

b) In this task, various artificial images were generated using regular periodic and non-periodic signals, starting with a periodic grid of impulses shown in Figure 2.



(a) Grid Impulses



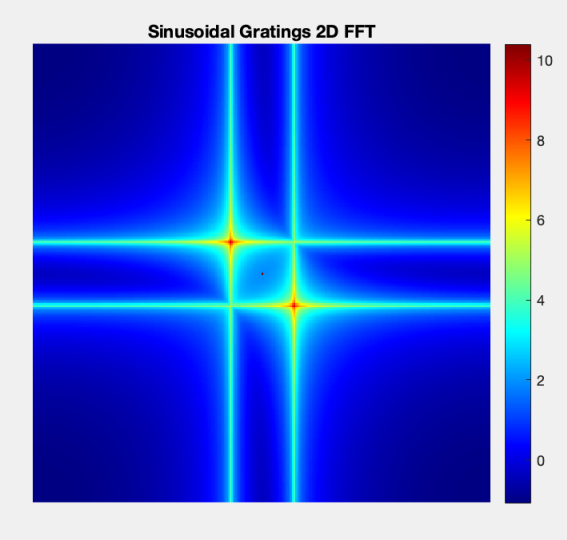
(b) FFT of Grid Impulses

Figure 2: Grid Impulses transformed with FFT

The FFT 2b plot is very similar to the image due to its periodicity. In this case, the FFT is conjugate symmetric; thus, the similarities in the pattern between time and frequency domains are very strong.

Continuing with the evaluation of periodic signals, sinusoidal gratings tilted at 45° are seen in Figure 3 below.

(a) Sinusoidal Gratings

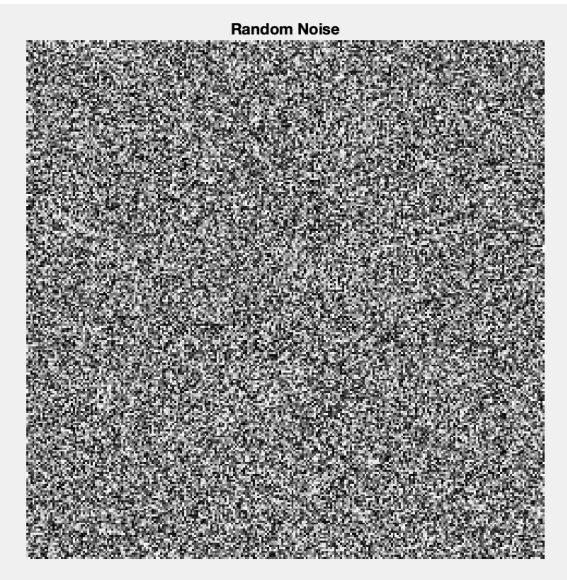


(b) FFT of Sinusoidal Gratings

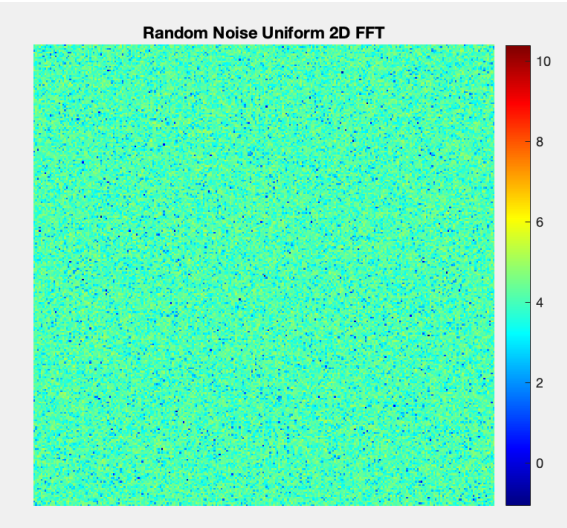
Figure 3: Sinusoidal Gratings at 45°

The distance of the two peaks from the origin in figure 3b represents the spatial frequency of the gratings, and their extended straight star pattern is due to the constant frequency across all gratings of the artificial image.

The first trial with a non-periodic artificial image was constructed with random noise as depicted in figure 4a.



(a) Random Noise



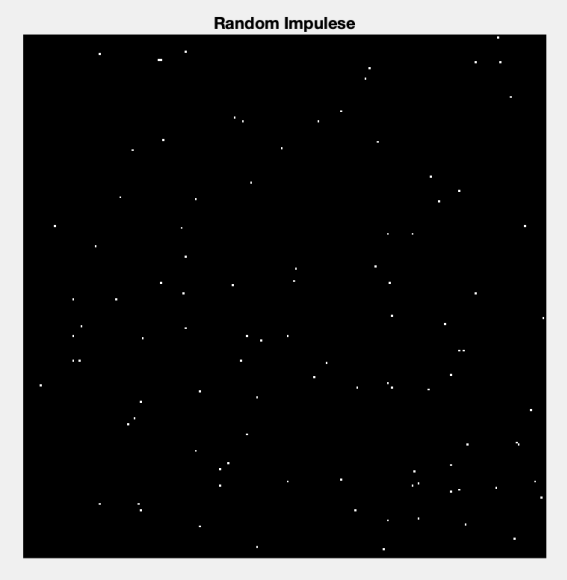
(b) FFT of Random Noise

Figure 4: Random Noise transformed with FFT

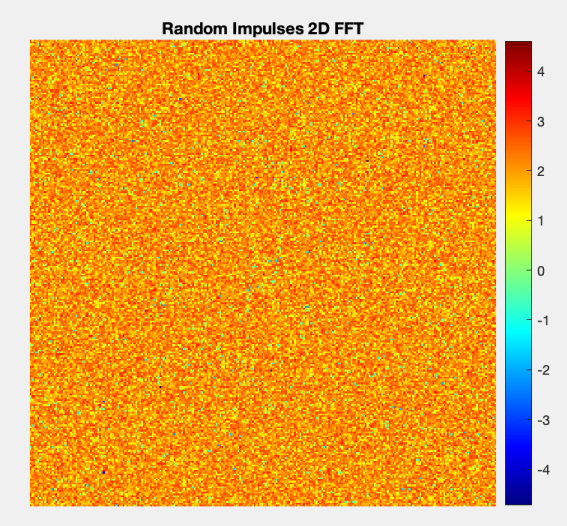
Due to random, uncorrelated pixel intensities, generated through the randn() MATLAB function that creates a random matrix of the specified image size, it is seen how all the frequencies are almost equally represented; thus, there are no specific peaks at any given point due to the uniform spread of noise.

Random impulses were also tested, to compare with the grid of impulses shown in the periodic

test images.



(a) Random Impulses

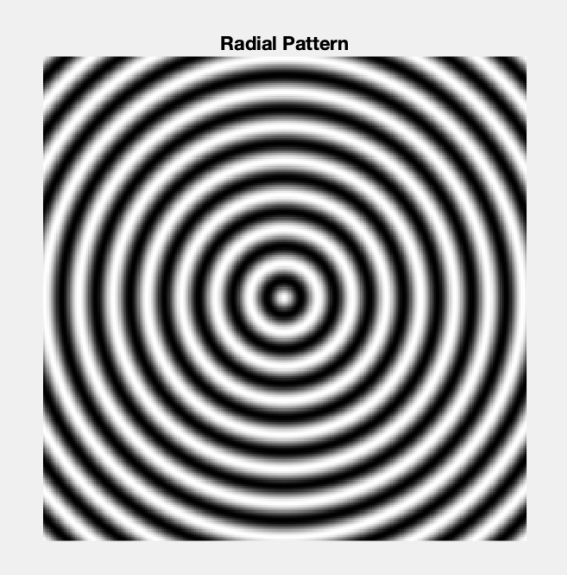


(b) FFT of Random Impulses

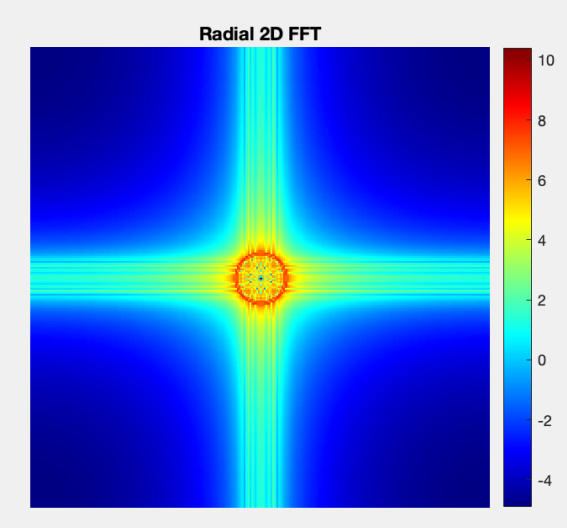
Figure 5: Random Impulses transformed with FFT

The similar reasoning as for random noise to the transform of the random impulses, though in this case, the impulses will have higher energy concentration. This occurs because although the impulses are few, they are non-periodic and a single impulse contributes evenly across frequencies in the Fourier domain.

The radial pattern shows a fixed representation of what looks like an image in figure 6a used by a magician to hypnotize someone.



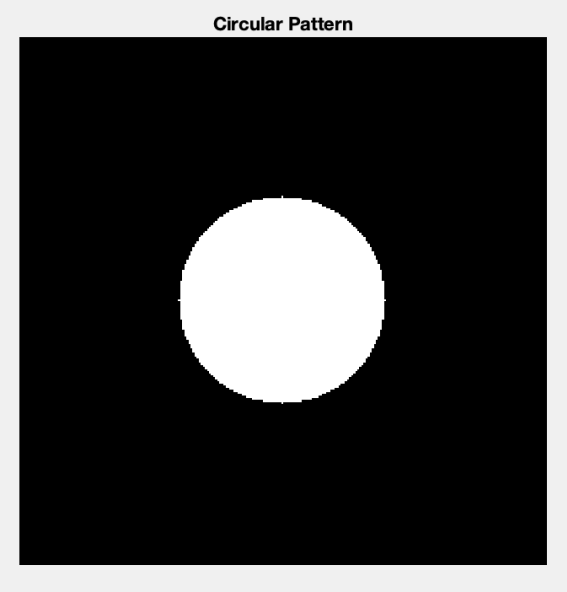
(a) Radial Pattern



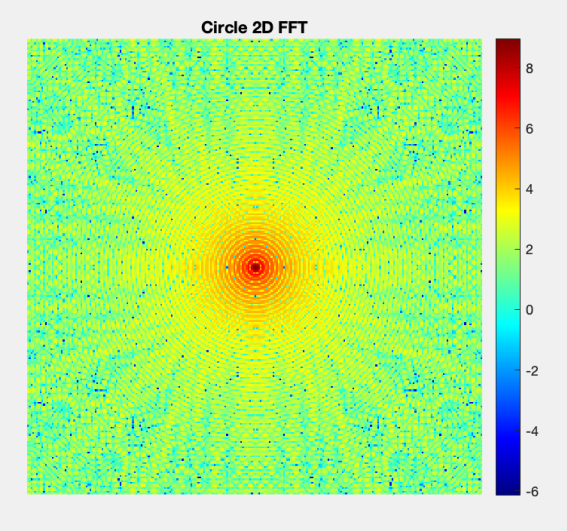
(b) FFT of Radial Pattern

Figure 6: Radial Pattern transformed with FFT

The frequencies here correspond to the radial variation. As with the sinusoidal gratings, the circle radius in 6b depends on the spatial frequency of the radial periodicity, preserving symmetry.



(a) Circular Pattern

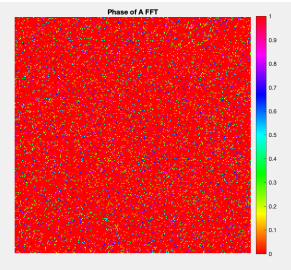


(b) FFT of Circular Pattern

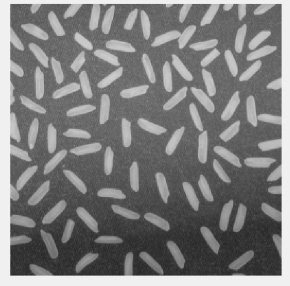
Figure 7: Circular Pattern Transformation Test

Concluding task (b) a circular pattern was tested as the last periodic artificial image with figure 7 above. Once again, periodicity displays symmetry in the Fourier domain, as 7b has a disk-like pattern where the size of the central point is inversely proportional to the circle radius of 7a.

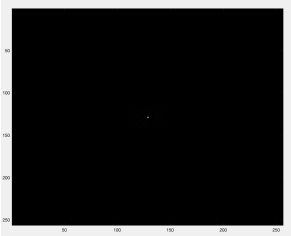
c) In task (c), the phase 8a of the cameraman 1 and amplitude 8c of the rice image 8b were computed and then added together to form figure 8d.



(a) Phase of Cameraman FFT



(b) Rice MATLAB Image



(c) Amplitude of Rice FFT

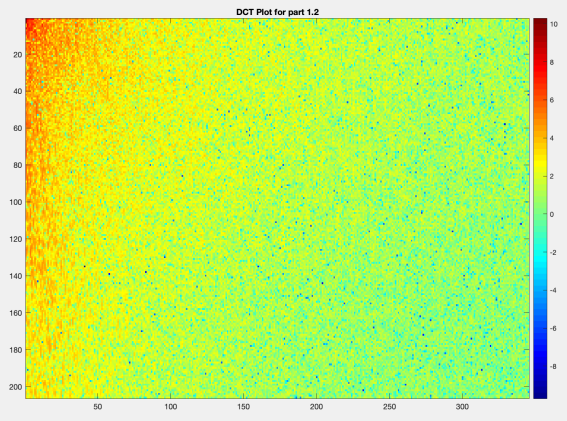
(d) Phase + Amplitude

Figure 8: Plots for task 1c

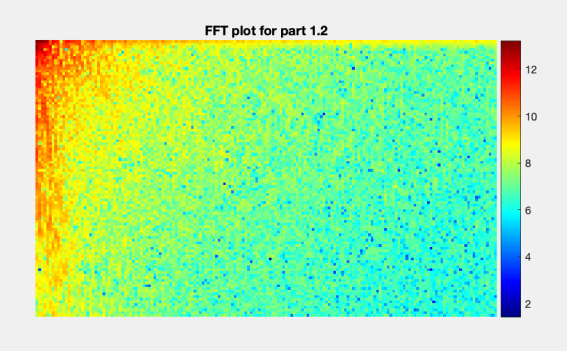
This result shown in 8d represents the combination of the cameraman's phase with the rice's amplitude. Evidently, the cameraman image is still discernible while not much visual evidence remains from the rice. Nevertheless, there is a noteworthy cloud-like pattern overlaid onto the cameraman. This happens because phase determines the spatial arrangement of frequency components, while amplitude indicates how strong in magnitude a particular frequency is. Therefore, phase will logically retain an observable image, but amplitude cannot reconstruct any details of the image by itself.

3.1.2 Discrete Cosine Transform (DCT)

a) The DCT and FFT plots of the Autumn image are shown below in figure 9.



(a) DCT of Autumn Image



(b) FFT of Autumn Image

Figure 9: DCT and FFT of Autumn Image

b) The resulting plots of DCT 9a and FFT 9b are very similar, showing most of the energy concentrated in the upper left corner, where the lower frequency coefficients lay. The DCT exhibits superior energy compaction, concentrating more information into fewer coefficients, which makes it ideal for applications like image compression. In contrast, the FFT retains phase information and handles complex values, offering broader applicability in tasks such as filtering and signal reconstruction.

3.1.3 Hadamard Transform

c) The 2D Hadamard transform of the rice image is shown in figure 10.

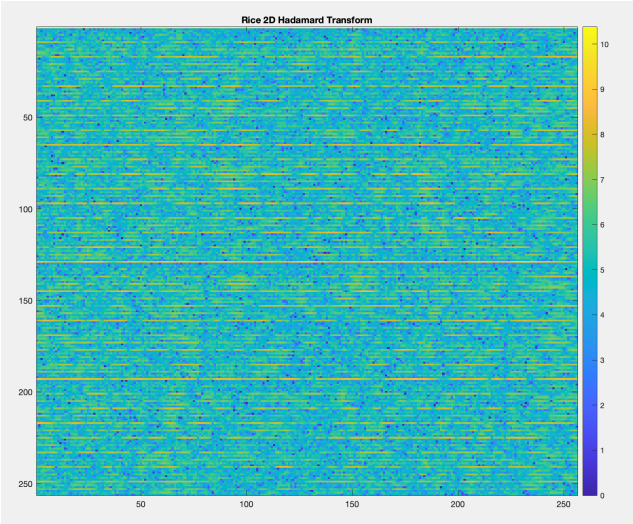


Figure 10: 2D Hadamard Transform of Rice Image

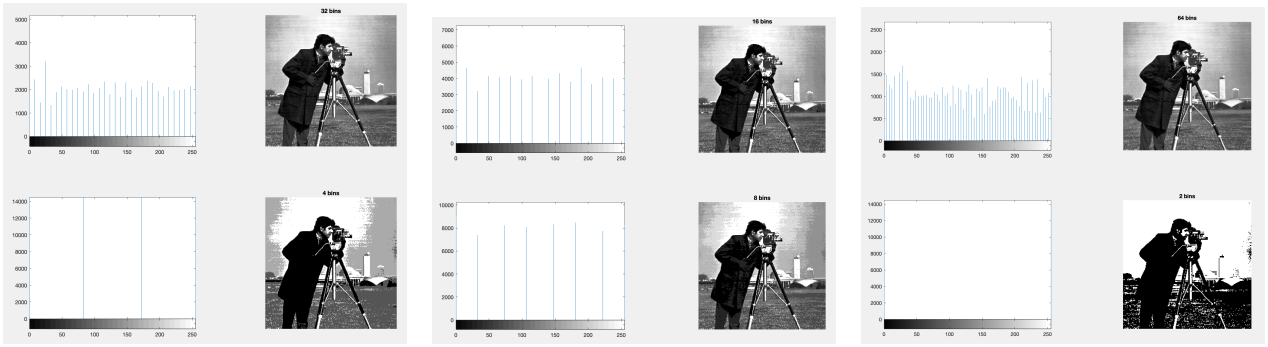
The Hadamard transform provides energy compaction for signals with simple or sparse structures, making it effective for compression in such cases. By concentrating most of the signal energy into a few coefficients, it minimizes storage requirements while preserving essential information. Its orthogonality ensures minimal redundancy among coefficients, enabling computational efficiency and making it useful for feature extraction and dimensionality reduction. Additionally, the recursive structure of the ordered Hadamard matrix supports fast and scalable computations, which are critical in applications like signal processing and image analysis.

3.2 Part II: Image Enhancement

In this section, the qualities of manipulating an image through its histogram will be shown to enhance some of the picture qualities, as well as edge detection and noise reduction.

3.2.1 Histogram Equalization

In this part, histogram equalization was applied to the cameraman image with varying bin counts, demonstrating how bin numbers influence dynamic range and contrast.



(a) 32-bins vs 4-bins (b) 16-bins vs 8-bins (c) 64-bins vs 2-bins

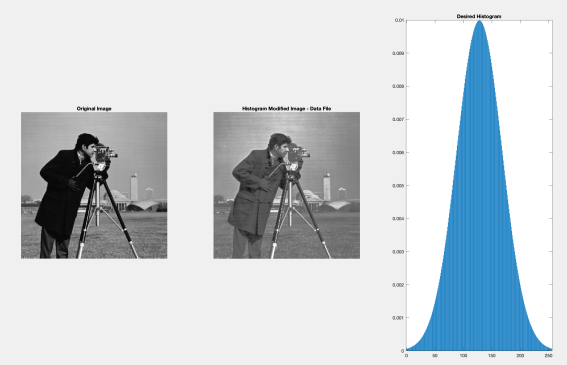
Figure 11: Cameraman for different number of bins in Histogram Equalization

It is clear from the figures in 11 how increasing the number of bins raises the dynamic range of the BW image. Therefore, each pixel will have the ability to represent a wider range of intensities. On 2 bins, for example, the picture is binary, as pixels can only be either black 0, or white 255. On the other hand, for 64-bins, the image contains the smoothest transitions in colors and shades, due to the higher dynamic range.

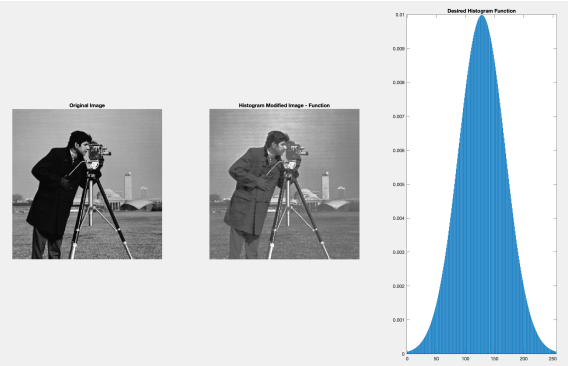
3.2.2 Program for Intensity Modification through Histogram

In this part, a program was created with the ability to modify the histogram intensity of the image. The workflow of the code is shown below:

1. Assume Input contains histogram with: intensity and probability
2. Normalize probabilities
3. Compute desired CDF (Cumulative Distribution Function) from probabilities
4. Compute CDF of input image
5. Map intensities using inverse of desired CDF
6. Apply mapping to image



(a) Histogram matching using data-file with Gaussian distribution



(b) Histogram matching using Gaussian function

Figure 12: Tree Edge Detection in Rotation

In figure 12 the two sub-figures display the cameraman picture on the right matched to a histogram with Gaussian distribution as displayed on the right-most curve plot. In 12a a file with values of such distribution was created, while for 12b the distribution was designed as a function. Both sub-figures display the same matched image, which has little dynamic range and contrast since middle (gray) values are accentuated at the expense of values closer to the edges that represent deeper black and white. The program to process a function input for histogram matching is below.

1. Sample the input function in 256 equally spaced samples
2. Normalize and keep as desired probabilities
3. Compute the CDF from the above probabilities
4. Map intensities using inverse of desired CDF
5. Apply mapping to image

3.2.3 Edge Generation

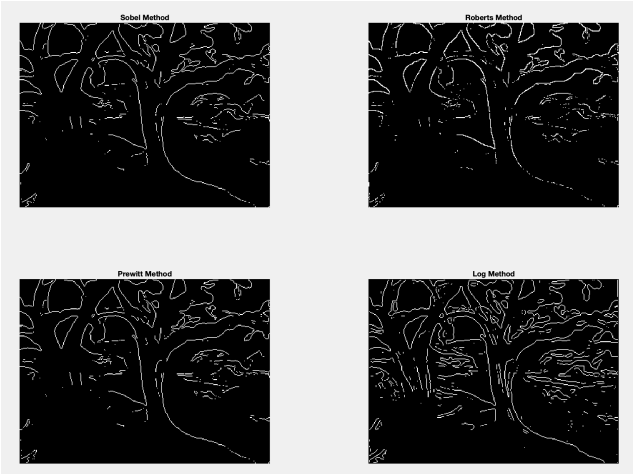
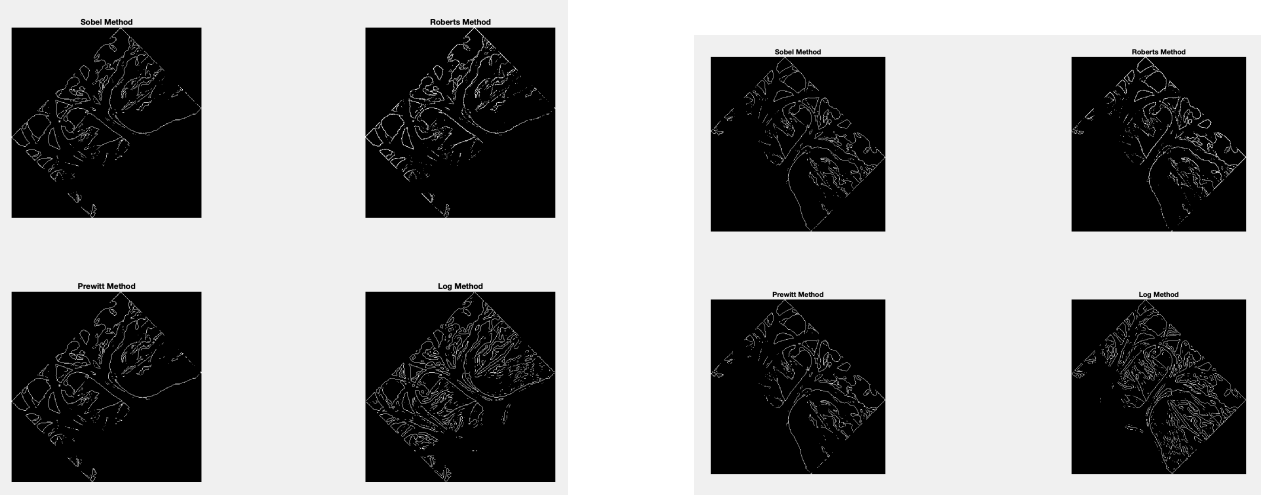


Figure 13: Four different edge detection methods for tree picture

In this part four different edge detection methods were implemented, as seen in figure 13 using the following methods: Sobel, Roberts, Prewitt and LoG (Laplacian of Gaussian. The LoG method

proved to discern the best edge detail, with Prewitt and Sobel close together in second place, while Roberts had the worse performance with the least detail. The reason LoG method outperformed others could lay at the fact that it combines smoothing to reduce noise with precise edge detection using the second derivative, making it very effective in capturing finer details.

The same exact test was performed for the tree picture tilted 45° each direction, as shown in figure 14.



(a) Trees Edge Detection Tilted +45°

(b) Trees Edge Detection Tilted -45°

Figure 14: Tree Edge Detection in Rotation

Even at tilted angles, the edge detection algorithms perform similarly, as the ranking remains the same as discussed previously for all methods.

3.2.4 Filtering with Various Noise

In this part, the autumn MATLAB image was used to evaluate the performance of de-noising salt & pepper and Gaussian noise through median filtering, using various block sizes.

(a) Median Filtering with 3x3 block size

(b) Median Filtering with 2x2 block size

(c) Median Filtering with 5x5 block size

Figure 15: Median filtering on salt & pepper noise

As portrayed above in 15, the salt & pepper noise is quite effectively eliminated using median filtering, however, to choose the ideal kernel size of the filter, an image quality must be sacrificed, either in sharpness or noise. For example, the 5x5 15c kernel wipes out most noise, but it induces blurring on the desired picture. On the other hand, a 2x2 15b kernel retains some noise but the image has more clarity. The 3x3 15a is the optimal balance between noise reduction and sharpness.

The same test was performed for Gaussian white noise with the results shown in figure 16.

(a) Median Filtering with 3x3 block size

(b) Median Filtering with 2x2 block size

(c) Median Filtering with 5x5 block size

Figure 16: Median filtering on Gaussian noise

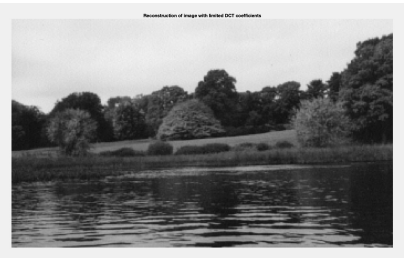
Similarly for Gaussian noise, the 5x5 16c adds too much blur, although the 3x3 16a also fails to reduce much noise while still blurring it. Overall, none of the three filters were very effective in noise reduction, since the noise was simply blurred but remained, thus not enhancing the image.

Further testing was performed with Poisson and Speckle noise, but the results were very similar to the Gaussian noise tests, thus for the sake of space were not included in this report.

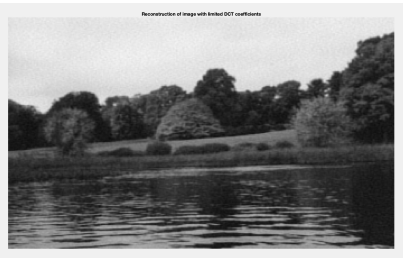
3.3 Part III: Image Compression

In this section, the DCT 2.1.2 method was used for compressing the autumn image, as shown in the previous section, with varying thresholds. The way compression works using DCT is first the transform coefficients are computed, a threhsold is chosen to zero-out insignificant coefficients, creating a representation of the image with fewer coefficients, but still retaining most information. Then, the inverse DCT allows the reconstruction of the compressed image and its qualities are evaluated.

Figure 17 below shows the ability of DCT to compress the autumn image for thresholds: 5, 10, 20.



(a) T = 5



(b) T = 10

(c) T = 20

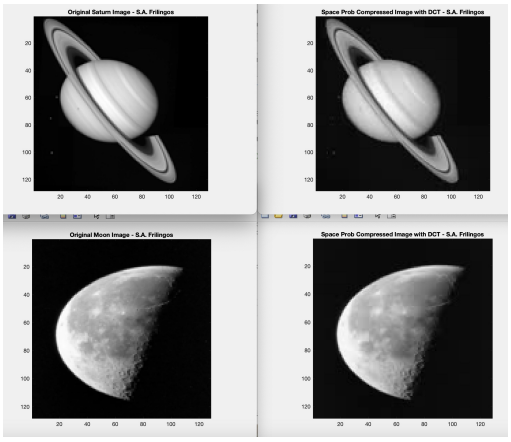
Figure 17: DCT compression with different thresholds T

The DCT concentrates the majority of its energy into the first few coefficients, enabling it to reconstruct a reasonably accurate image even with higher threshold values. However, as expected, noticeable degradation occurs as the compression threshold T increases, as shown in 17a to 17c.

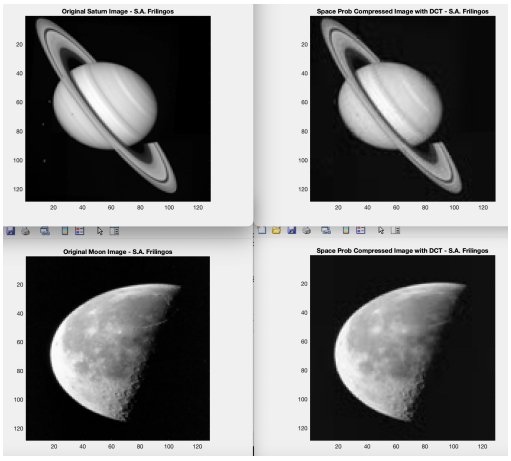
Simulations were conducted using various thresholds and block sizes ranging from 2x2 to 32x32. Results showed that smaller block sizes yielded better image analysis and finer detail preservation. However, the threshold parameter needed to be adjusted in proportion to the block size. Smaller blocks required lower thresholds for accurate reconstruction, while larger blocks achieved similar reconstruction quality at higher thresholds, making them more efficient for compression.

3.4 Part IV: Design Exercise

In this design scenario, MATLAB images of Saturn and the Moon were transmitted through a space probe, contaminated by Gaussian noise. Each frame is sent with a 4:1 compression, using 8x8 and 16x16 DCT block sizes. The goal is to design the optimal choice of compression parameters of block size and threshold, for both noise and noiseless scenarios.

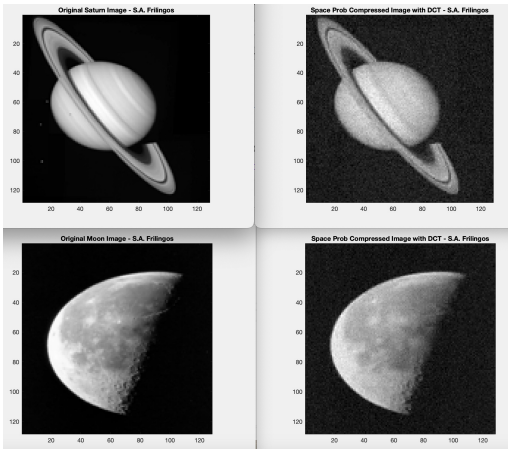


(a) 8x8 Noiseless 4:1 compression

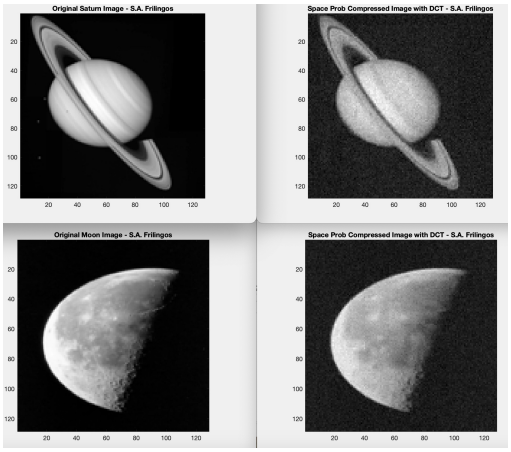


(b) 16x16 Noiseless 4:1 compression

Figure 18: Reconstruction with 4:1 Compression



(a) 8x8 Gauss Noise, T=$3\sigma_N$, $\sigma_N = 50$



(b) 16x16 Gauss Noise, T=$3\sigma_N$, $\sigma_N = 50$

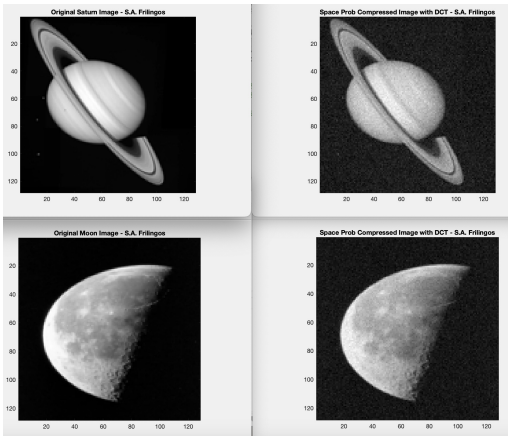
Figure 19: Reconstruction with T=$3\sigma_N$

Figure 18 and 19 reconstructed the images on the right column of each subfigure with the specified parameters. Table 1 below shows the mean squared error and Peak SNR for each of these tests in order to evaluate performance.

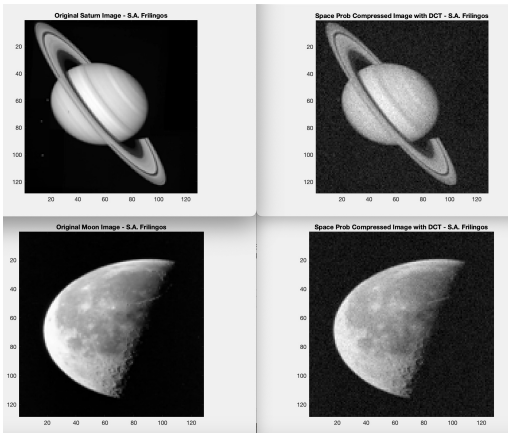
Image	MSE	PSNR (dB)	Block Size
Saturn 18a	5.67	40.60	8
Moon 18a	7.06	39.64	8
Saturn 18b	6.96	39.70	16
Moon 18b	9.38	38.40	16
Saturn 19a	66	29.94	8
Moon 19a	66.50	29.90	8
Saturn 19b	66.54	29.90	16
Moon 19b	67.31	29.85	16

Table 1: Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR) for 18 and 19

On multiple runs of the noisy cases, 8x8 showed a slightly lower MSE than 16x16, settling in the range between 64-66 while 16x16 ranged around 66-70. However, the PSNR remained almost the same for both cases at around 30. Visually, both results are quite pleasing to the eye, since the noise with variance σ_N^2 is noticeable but not devastating to the pictur quality. Still, Saturn's three moons have been lost. The threshold at T=$3\sigma_N$ proved to work quite well. Nevertheless, it could be interesting to test for a few different thresholding values, as that is a fundamental parameter in the reconstruction quality, which could lead to a smaller MSE and a higher PSNR.



(a) 8x8 with Gauss Noise, T=σ_N, $\sigma_N = 50$



(b) 16x16 with Gauss Noise, T=σ_N, $\sigma_N = 50$

Figure 20: Test with smaller threshold T=σ_N

The sub-figures displayed in figure 20 display the results of the reconstructed images using a smaller threshold, for both block sizes 8x8 and 16x16, with table 2 below showing the quantitative performance results.

Image	MSE	PSNR (dB)	Block Size
Saturn 20a	52.48	30.93	8
Moon 20a	53.51	30.85	8
Saturn 20b	53.17	30.87	16
Moon 20b	54.45	30.77	16

Table 2: Mean Squared Error (MSE) and Peak Signal-to-Noise Ratio (PSNR) for 20

As was assumed, lowering the threshold showed a decrease in MSE by about 10, while PSNR increased only ever so slightly. Thus, an overall better reconstruction performance against noise

was shown. Using 8x8 block size had slightly better MSE once again, compared to 16x16. Also, with T=σ_N and 8x8 block size, the three moons of Saturn remain visible after noise. Therefore, my optimal recommendation is to use a threshold of T=σ_N and block size 8x8, where σ_N refers to the noise variance.

3.5 Part V: Image Restoration

The last section pertained the development of two techniques for restoring an image that is corrupted with noise and blurring: Inverse Filtering 2.5.1 and Wiener Filtering 2.5.2.
First, the program sought to create a degraded image, using the well-known cameraman. The list below shows the steps taken in the code:

1. Create 5x5 and 7x7 blur kernels with $\sigma^2 = 1$
2. Use imfilter() to convolve across the entire cameraman image
3. Use BSNR=20dB
4. Compute noise variance:
   (a) Find variance of blurred image
   (b) Find noise variance of BSNR
5. Finally, use imnoise() to get the degraded image using the computed noise variance

The resulting degraded images for the two kernel sizes are shown below in figure 21.

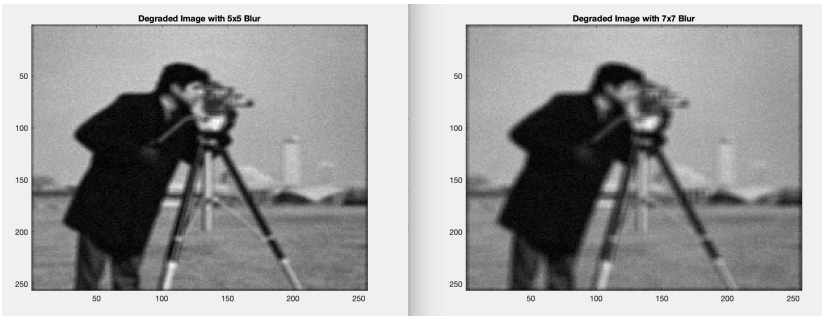


Figure 21: Degraded images with 5x5 (left) and 7x7 (right) kernel blur and BSNR=20dB

3.5.1 Inverse Filtering

Below is a short analysis of the code designed to implement inverse filtering in order to retrieve the original image from the degraded version of figure 21.

1. Convert point-spread function of the 5x5 and 7x7 blur kernels to an optical transfer function (H_5, H_7)
2. Compute the FFT of degraded image
3. Apply inverse filtering 2.5.1
4. Take the inverse FFT of the real part and plot

The results of the inverse filtering are portrayed in figure 22. It is hard to discern significant differences. Inverse filtering works best for noiseless scenarios. In this case, the noise is slightly accentuated with the filtering, however some sharp edges, such as the ones around the camera, become more pronounced, thus the restored image is showing improved sharpness.

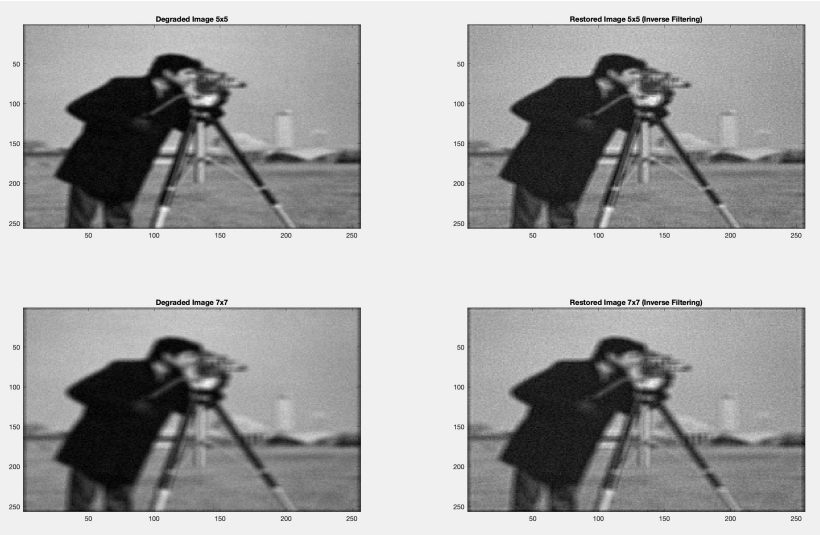


Figure 22: Inverse filtering results on right column, with 5x5 (top) and 7x7 (bottom)

3.5.2 Wiener Filtering

To create the program for this filter, the following steps were taken:

1. Form a regularizing term = $\frac{\sigma_N}{\sigma_b}$
2. Apply Wiener filtering through MATLAB's 'deconvwnr()' built-in function
3. Apply Gaussian filtering with $\sigma^2 = 1$ to improve the contrast in the restored image using MATLAB's 'imgaussfilt()'
4. Plot the result

The restored image using Wiener filtering (23) shows significantly better detail compared to inverse filtering, particularly in the edges around the subject, making it more effective for deblurring. Wiener filtering's ability to account for noise and optimize restoration based on statistical properties results in better clarity and sharpness without adding excessive noise. However, some noise is still introduced across the smooth regions of the image.

Artifacts or square patterns near the borders of the image in the Wiener-filtered result could be due to boundary effects, where the filtering process struggles with limited or mismatched information at the edges. This may arise from padding methods or an incomplete model of noise characteristics at the image borders.

It is also worth noting that inverse filtering performs better for relatively noiseless degradation, as it directly applies the inverse of the degradation function without accounting for noise. However, in the presence of noise, inverse filtering amplifies these unwanted signals, leading to poor restoration quality, particularly in smooth regions.

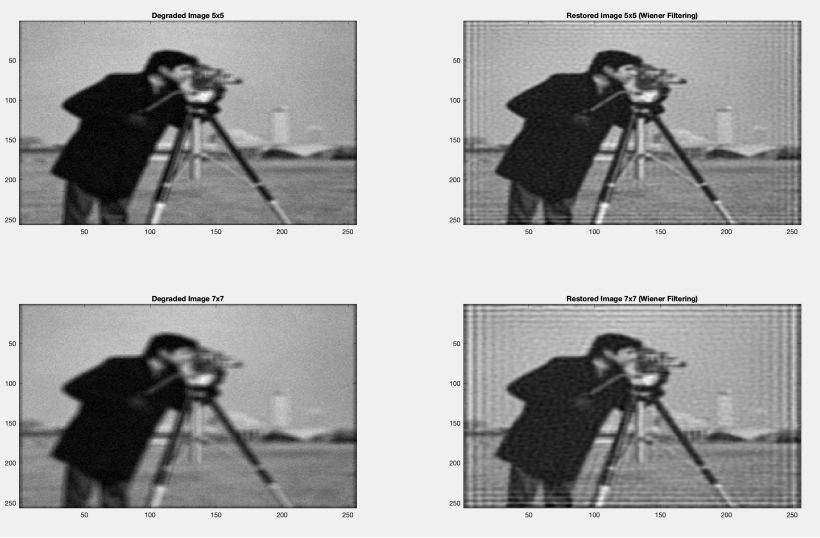


Figure 23: Wiener filtering results on right column, with 5x5 (top) and 7x7 (bottom)

4 Results Discussion

In this section, an analytical discussion will go through the results obtained throughout the experimental process in order to provide an interpretation of the results and findings.

4.1 Image Transforms

In the first part of the experiment, the FFT, DCT and Hadamard transforms were tested in producing representations of 2D images. In the FFT of an image, it was observed how the majority of the coefficient energy lies in the smaller frequencies, which refer to the smoother picture images with no sharp edges or rapid changes. Furthermore, the FFT representations of artificial images were compared for periodic and non-periodic signals, where it was shown how for periodic ones there was a consistent symmetry in the Fourier domain. Concluding the FFT tests, an image was formed through the summation of the phase and magnitude of two different images. This process revealed how the phase retains most of the spatial information of the picture, thus was evident in the final result, while the amplitude did not retain any visually observable information from the original rice image.

Moving forwards with the DCT, there are many similarities with FFT with most energy concentrated in the upper left corner, representing lower frequency components. Both transforms effectively capture the image's frequency characteristics, but the DCT demonstrates better energy compaction, making it ideal for image compression like JPEG. The FFT, however, retains phase information and handles complex values, offering broader applicability in tasks like filtering and signal analysis. While visually comparable, the DCT's efficiency in concentrating information into fewer coefficients gives it an edge for compression-focused applications.

The Hadamard transform is computationally efficient and faster than FFT and DCT, making it suitable for processing images with sharp transitions or binary data. Its simplicity and reliance on binary operations reduce computational overhead, making it advantageous for low-complexity image analysis in software applications. However, its energy compaction is less effective than the DCT, limiting its utility for tasks like compression.

4.2 Image Enhancement

In Part II, histogram manipulation was demonstrated to enhance images by adjusting dynamic range. Reducing the number of bins increased contrast but limited the range of a BW image 11. A program was also developed to modify image intensity based on histogram probabilities using a data file or a function as input. The test with a Gaussian histogram input showed a faded result, with more 'hazy' grays, due to a boost of the middle pixel values.

Edge detection tests on the MATLAB tree image showed 13 the Laplacian of Gaussian (LoG) outperformed others due to its noise robustness, combining smoothing with precise edge detection via the second derivative. Tilting the image 14 did not affect algorithm performance, confirming rotational invariance.

Median filtering was evaluated for noise reduction on salt & pepper 15 and Gaussian noise 16. Larger block sizes over-blurred the image, while smaller ones failed to reduce noise effectively. The optimal block size for sharpness and noise elimination was 3x3. However, for Gaussian noise, median filtering was ineffective, increasing blurring while retaining noise due to the uniform and dense distribution of Gaussian noise compared to the impulsive salt & pepper.

4.3 Image Compression

In the image compression experiments, the DCT demonstrated its effectiveness in reducing the number of coefficients while retaining most of the image information. As the threshold T increased from 5 to 20 17, more coefficients were zeroed, resulting in greater compression but with noticeable degradation in image quality. This trade-off reflects the balance between compression efficiency and image fidelity, with lower thresholds preserving finer details and higher thresholds prioritizing compression.

Additionally, simulations with varying block sizes revealed that smaller blocks preserved image details more effectively due to localized analysis. However, smaller blocks required lower thresholds for accurate reconstruction, making them less efficient for compression. Conversely, larger blocks allowed higher thresholds with minimal quality loss, proving more efficient for compression tasks. This is due to their ability to compact more information into fewer coefficients while maintaining acceptable reconstruction quality.

4.4 Design Experience

In this study, images of Saturn and the Moon (128x128 pixels) were compressed and reconstructed under noiseless and noisy conditions using DCT with 8x8 and 16x16 block sizes. A 4:1 compression ratio was applied, meeting the bandwidth requirement. Results showed that smaller blocks (8x8) offered better MSE and PSNR, particularly in noisy scenarios, due to finer localized analysis. Larger blocks (16x16) achieved similar PSNR but reduced precision in noise handling, as they compacted more gobal information.

In noisy conditions, Gaussian noise affected both block sizes. A threshold of $T = \sigma_N$ provided an optimal trade-off between noise suppression and detail preservation, with 8x8 blocks retaining features like Saturn's moons. Lower thresholds improved MSE slightly but had diminishing visual benefits. For noiseless cases, both block sizes performed comparably, though 8x8 still preserved finer details. The recommendation is to use 8x8 blocks and a threshold of $T = \sigma_N$ for visually acceptable results across scenarios.

4.5 Image Restoration

The program restored degraded images using inverse and Wiener filtering under 5x5 and 7x7 Gaussian blur kernels with 20 dB noise. Inverse filtering showed limited effectiveness in noisy scenarios, as it amplified noise significantly in smooth regions. This is because inverse filtering does not account for noise, instead focusing solely on reversing the degradation function. The results for the 7x7 kernel revealed more pronounced noise amplification compared to the 5x5 kernel due to the broader blur spreading noise across a larger region.

Wiener filtering, on the other hand, provided much better noise suppression and detail restoration. Its ability to incorporate noise and image statistics enabled sharper edges and reduced noise in both kernel cases. The 5x5 kernel produced finer detail and sharper reconstruction than the 7x7 kernel, as the latter introduced slight smoothness. However, Wiener filtering introduced square-like artifacts near the borders, likely due to boundary effects or incomplete modeling of noise characteristics in those regions.

5 Conclusion

This lab explored several key image processing techniques, each with unique strengths and limitations. In the image transforms 3.1 section, the FFT, DCT, and Hadamard transforms were tested. The FFT highlighted energy concentration in lower frequencies and demonstrated the critical role of phase information in retaining spatial image details. The DCT offered better energy compaction, making it ideal for compression, while the Hadamard transform excelled in computational efficiency but lacked effective energy compaction.

In image enhancement 3.2, histogram manipulation showed how adjusting bins impacts dynamic range and contrast. The LoG outperformed other edge detection algorithms due to its robustness against noise and rotational invariance. Median filtering was effective for salt-and-pepper noise but struggled with Gaussian noise due to its dense distribution.

For image compression 3.3, DCT experiments revealed a trade-off between thresholds and block sizes. Smaller blocks preserved finer details but required lower thresholds, while larger blocks allowed higher thresholds with minimal quality loss, favoring compression efficiency.

In the design experience 3.4, images of Saturn and the Moon were compressed under noisy and noiseless conditions. Smaller blocks (8x8) performed better in noisy cases due to localized analysis, while larger blocks (16x16) compacted information efficiently but struggled with noise.

Finally, in image restoration 3.5, inverse filtering was limited in noisy scenarios, amplifying noise, while Wiener filtering provided superior restoration by accounting for noise and image statistics. However, Wiener filtering occasionally introduced border artifacts due to boundary effects.