EXPERIMENT 7S: Image Processing

Student: Haoxiang Huang

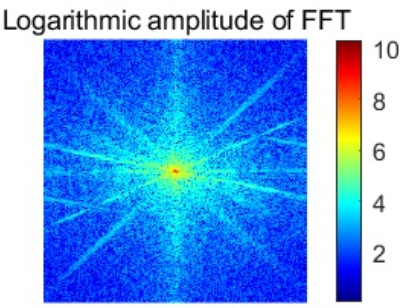
CID: 02470313

Demostrator: Siying Liu

Part A: Image transformation

1. FFT

Part 1

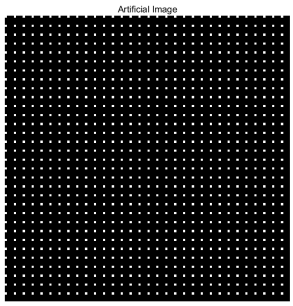


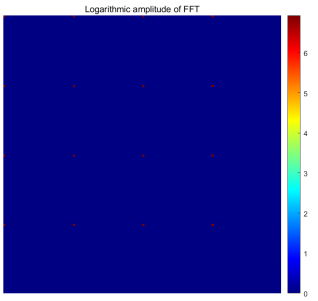
LHS is the original image, RHS is the log amplitude of FFT (shifted to the middle).

We can find that the low index usually possess higher values compared to the coefficients with high index. This is because the natural images

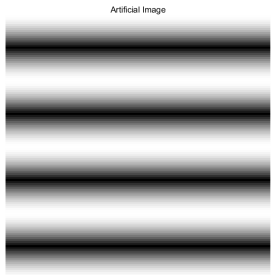
are very smooth.

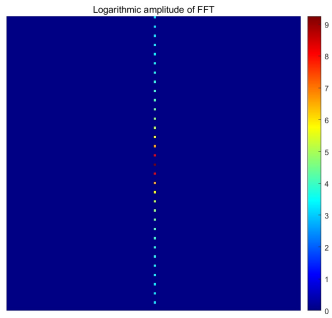
Part 2



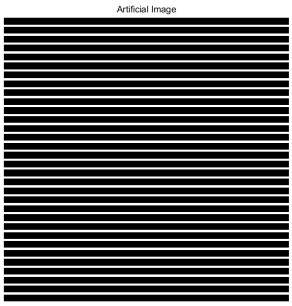


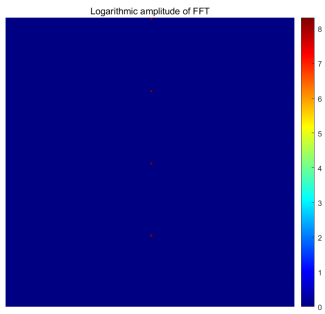
(1)



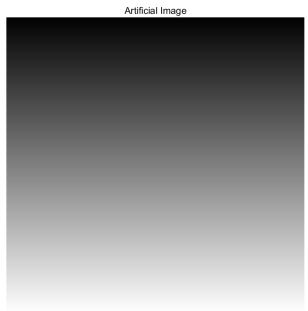


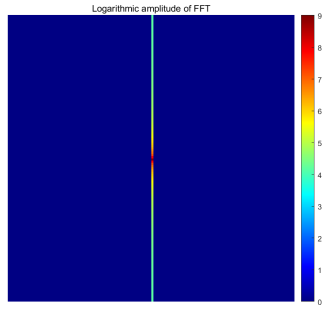
(2)





(3)





(4)

(1) the Artifical image of a grid of impluses shows the DFT is also impluse-like

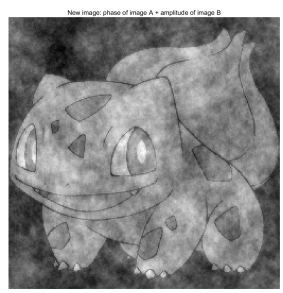
(2)(3) is the periodic image, their DFT is discrete.

(4) is the non perdiodic image, its DFT is continous

Part 3

original

Phase A + amplitude B

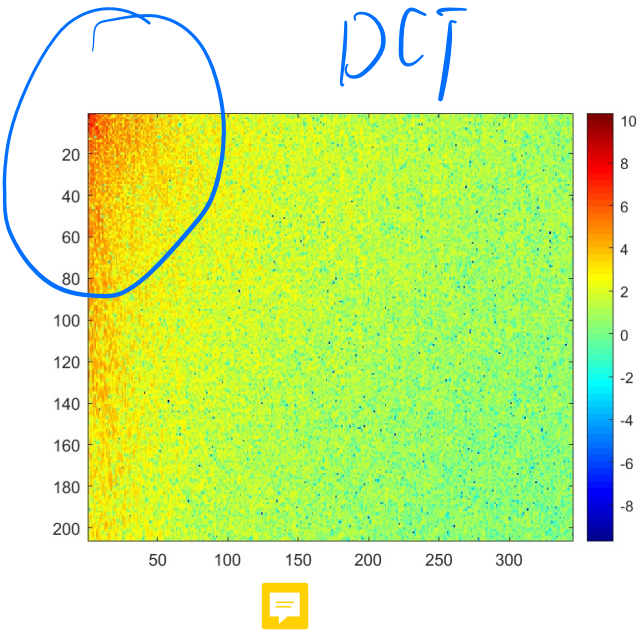


The phase of image indicates the distribution of frequency of the image. The high frequency compents of the images are preserved by the phase. Therefore, we can recogenize the image contant based on phase.

2. DCT

original

DCT

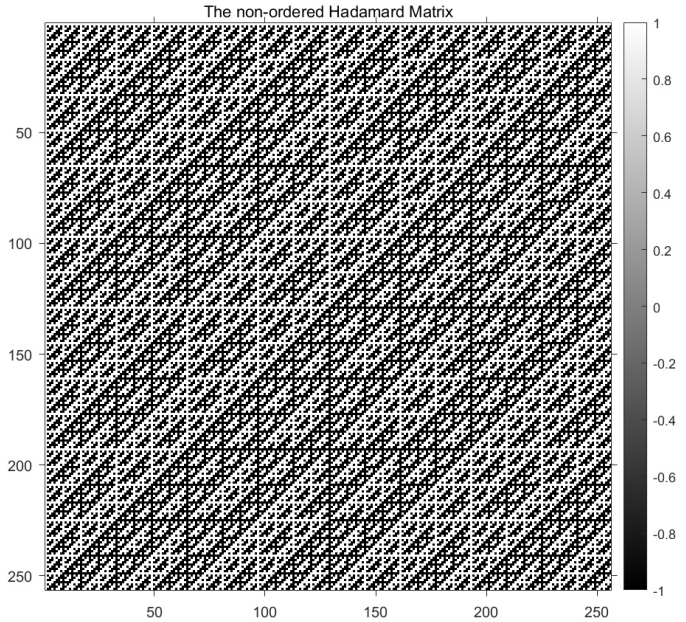


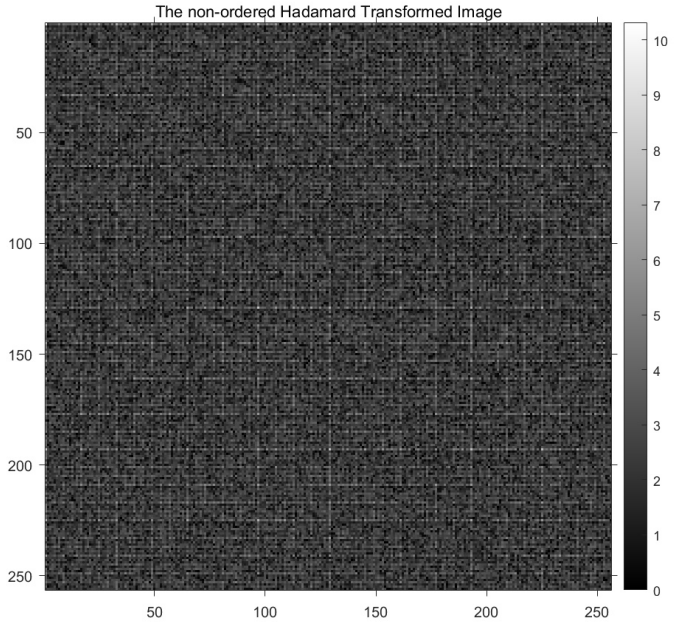
DCT low frequency components are located at the top-left corner. while FFT are located at four corners.

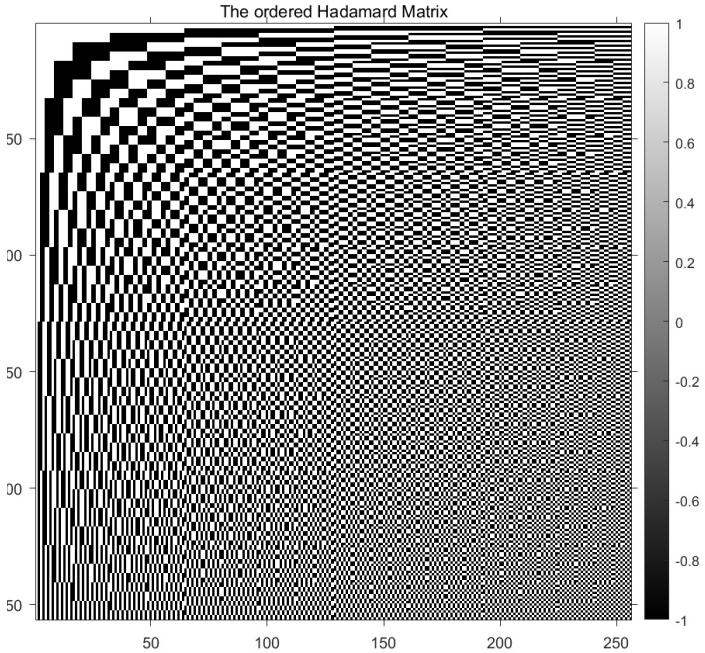
3. Hadamard

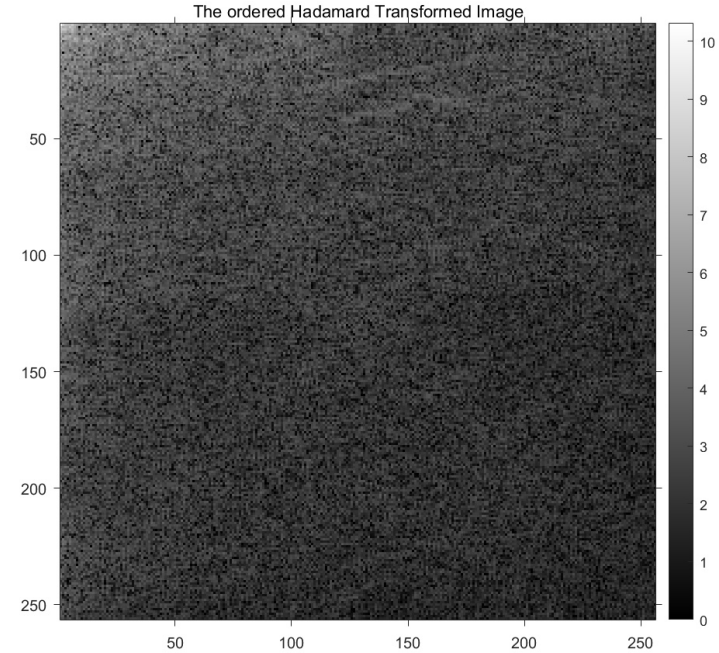
Pros: 1. Computational efficiency 2. Fast Algorithm like Fast-FFT

3. Orthogonality of basic vector 4. Noise Resilience



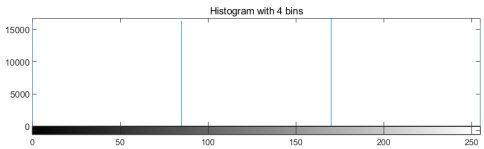


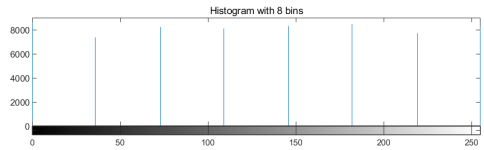


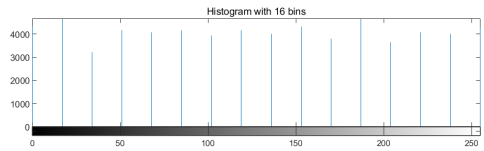


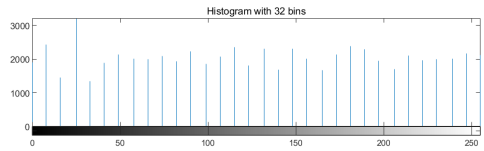
Part B Image Enhancement

1. Histogram Equlisation



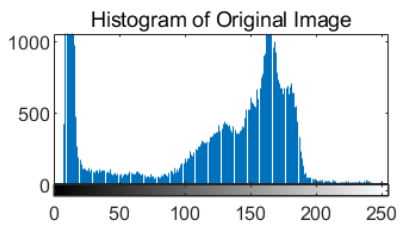




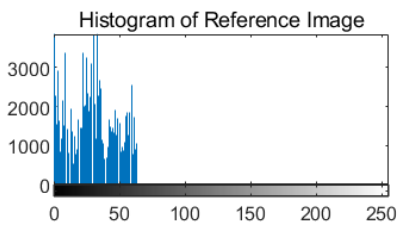


Part2

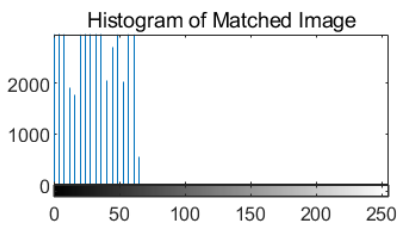
Original Image (Cameraman)



Reference Image (Kids)



Histogram Matched Image



we can see the result hist of matched image is similar to the reference image.

Part3 edge detection

Customed 45° sobel mask

$$\begin{pmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{pmatrix}$$

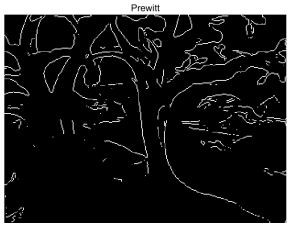
Customed -45° Sobel mask

$$\begin{pmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{pmatrix}$$

Sobel

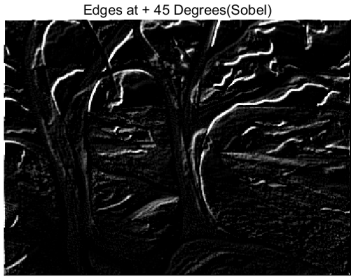
Roberts

Prewitt

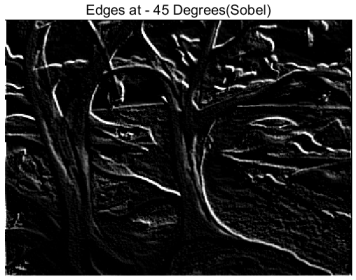


Log

Edges at + 45 Degrees(Sobel)



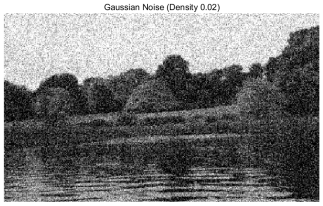
Edges at - 45 Degrees(Sobel)



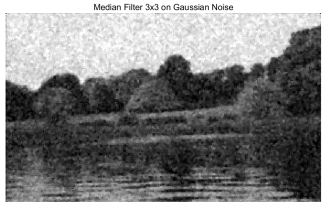
Edges at +/- 45 Degrees(Sobel)

Part 4 Median Filter

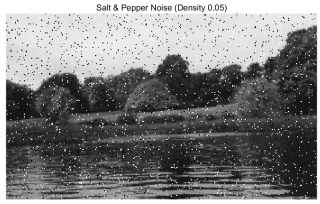
Gaussian Noise (Density 0.02)



Median Filter 3x3 on Gaussian Noise

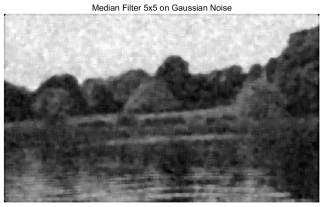


Salt & Pepper Noise (Density 0.05)

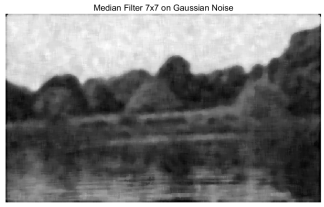


Median Filter 3x3 on Salt & Pepper Noise

Median Filter 5x5 on Gaussian Noise



Median Filter 7x7 on Gaussian Noise



Median Filter 5x5 on Salt & Pepper Noise

Median Filter 7x7 on Salt & Pepper Noise

we can find that the larger kernal, the result becomes more blur

Part C Image Compression

Threshold = 5

Threshold = 10

Threshold = 20

Threshold = 35

Threshold = 55

Threshold = 80

```
% Process each 8x8 block
for i = 1:blockSize:rows
    for j = 1:blockSize:cols
        % Extract the block
        block = I(i:min(i+blockSize-1, rows), j:min(j+blockSize-1, cols));

        % Apply DCT to the block
        J = dct2(block);

        % Apply the threshold
        J(abs(J) < threshold) = 0;

        % Perform IDCT
        block = idct2(J);

        % Place the processed block back into the image
        K(i:min(i+blockSize-1, rows), j:min(j+blockSize-1, cols)) = block;
    end
end
```

DCT coefficient less than threshold will be set to 0.

Change threshold and repeat above process, we can get the image above. We find the large threshold, the image is more blured.

Part D: Design Exercise

```
function K = fDCT_Tx(I, blockSize, compressedRatio)

    [rows, cols] = size(I);
    K = zeros(rows, cols);

    % Calculate the number of coefficients to keep based on the compressed ratio
    numCoefficients = blockSize * blockSize / compressedRatio;
    numCoefficientsPerDimension = sqrt(numCoefficients);

    for i = 1:blockSize:rows
        for j = 1:blockSize:cols
            % Extract the block
            block = I(i:min(i+blockSize-1, rows), j:min(j+blockSize-1, cols));

            % Apply DCT
            J = dct2(block);

            % Zero out all but the top left 'numCoefficientsPerDimension x numCoefficientsPerDimension' coefficients
            J(numCoefficientsPerDimension+1:end, :) = 0;
            J(:, numCoefficientsPerDimension+1:end) = 0;

            % Replace the block in the compressed image
            K(i:min(i+blockSize-1, rows), j:min(j+blockSize-1, cols)) = J;
        end
    end
```

Based on the compressed ratio, we will zero out all but top-left.

Rx- IDCT:

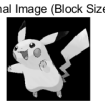
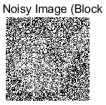
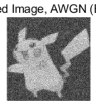
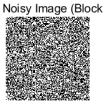
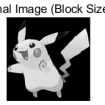
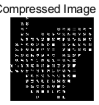
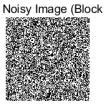
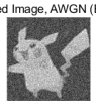
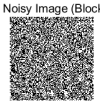
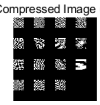
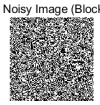
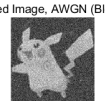
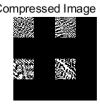
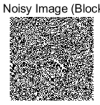
```
function K = fIDCT_Rx(J, blockSize)
    [rows, cols] = size(J);
    K = zeros(rows, cols);
    for i = 1:blockSize:rows
        for j = 1:blockSize:cols
            block = J(i:min(i+blockSize-1, rows), j:min(j+blockSize-1, cols));
            % Inverse DCT
            block = idct2(block);

            % Replace the block
            K(i:min(i+blockSize-1, rows), j:min(j+blockSize-1, cols)) = block;
        end
    end

    % Normalize the image
    K = mat2gray(K);
end
```

Results:

The compressed image by fDCT-Tx will through the 10dB AWGN channel, and received Image will be reconstructed by fIDCT-Rx using IDCT.

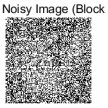
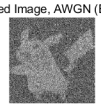
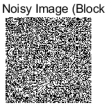
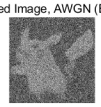
Images with Different Block Sizes (SNR: 10 dB)

Block Size

Original Image (Block Size: 2)	Transmitted Compressed Image (Block Size: 2)	Received Noisy Image (Block Size: 2)	Reconstructed Image, No Noise (Block Size: 2)	Reconstructed Image, AWGN (Block Size: 2)
				
Original Image (Block Size: 4)	Transmitted Compressed Image (Block Size: 4)	Received Noisy Image (Block Size: 4)	Reconstructed Image, No Noise (Block Size: 4)	Reconstructed Image, AWGN (Block Size: 4)
				
Original Image (Block Size: 8)	Transmitted Compressed Image (Block Size: 8)	Received Noisy Image (Block Size: 8)	Reconstructed Image, No Noise (Block Size: 8)	Reconstructed Image, AWGN (Block Size: 8)
				
Original Image (Block Size: 16)	Transmitted Compressed Image (Block Size: 16)	Received Noisy Image (Block Size: 16)	Reconstructed Image, No Noise (Block Size: 16)	Reconstructed Image, AWGN (Block Size: 16)
				
Original Image (Block Size: 32)	Transmitted Compressed Image (Block Size: 32)	Received Noisy Image (Block Size: 32)	Reconstructed Image, No Noise (Block Size: 32)	Reconstructed Image, AWGN (Block Size: 32)
				
Original Image (Block Size: 64)	Transmitted Compressed Image (Block Size: 64)	Received Noisy Image (Block Size: 64)	Reconstructed Image, No Noise (Block Size: 64)	Reconstructed Image, AWGN (Block Size: 64)
				

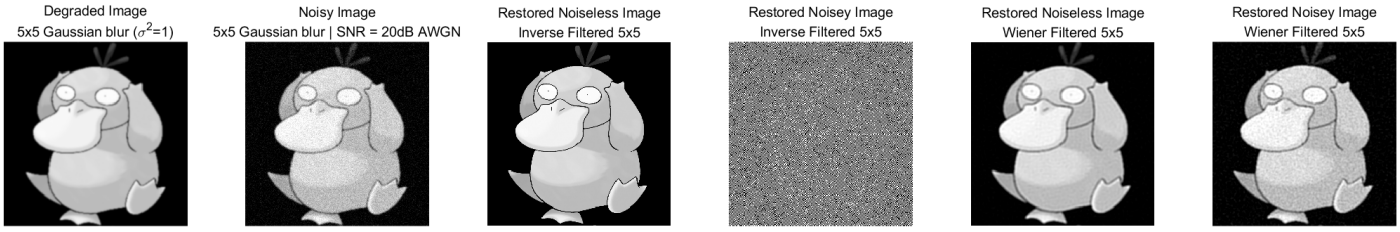
Images with Different Block Sizes (SNR: 0 dB)

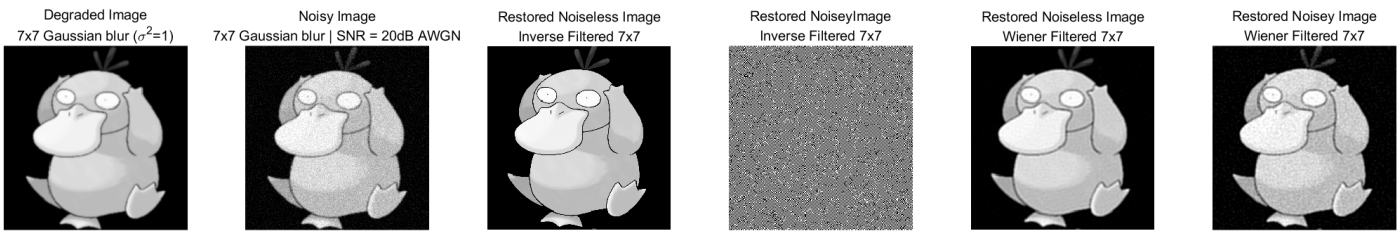
Block Size

Original Image (Block Size: 2)	Transmitted Compressed Image (Block Size: 2)	Received Noisy Image (Block Size: 2)	Reconstructed Image, No Noise (Block Size: 2)	Reconstructed Image, AWGN (Block Size: 2)
				
Original Image (Block Size: 4)	Transmitted Compressed Image (Block Size: 4)	Received Noisy Image (Block Size: 4)	Reconstructed Image, No Noise (Block Size: 4)	Reconstructed Image, AWGN (Block Size: 4)
				
Original Image (Block Size: 8)	Transmitted Compressed Image (Block Size: 8)	Received Noisy Image (Block Size: 8)	Reconstructed Image, No Noise (Block Size: 8)	Reconstructed Image, AWGN (Block Size: 8)
Original Image (Block Size: 16)	Transmitted Compressed Image (Block Size: 16)	Received Noisy Image (Block Size: 16)	Reconstructed Image, No Noise (Block Size: 16)	Reconstructed Image, AWGN (Block Size: 16)
Original Image (Block Size: 32)	Transmitted Compressed Image (Block Size: 32)	Received Noisy Image (Block Size: 32)	Reconstructed Image, No Noise (Block Size: 32)	Reconstructed Image, AWGN (Block Size: 32)
Original Image (Block Size: 64)	Transmitted Compressed Image (Block Size: 64)	Received Noisy Image (Block Size: 64)	Reconstructed Image, No Noise (Block Size: 64)	Reconstructed Image, AWGN (Block Size: 64)

PartE Image Restoration

Degraded Images with Different Filtering (SNR: 20 dB)





We can find that Inverse Filtered fails to process degraded image with AWGN. while Wiener Filtered Process it well.