BILKENT UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF ELECTRIC AND ELECTRICAL ENGINEERING

EE 391

SIGNALS AND SYSTEMS

Section 2

Mini Project 2

Ömer Oktay Gültekin

21901413

18.12.2023

1) M-point Averaging Filter

Apples.bmp file is chosen for the analysis throughout the project.

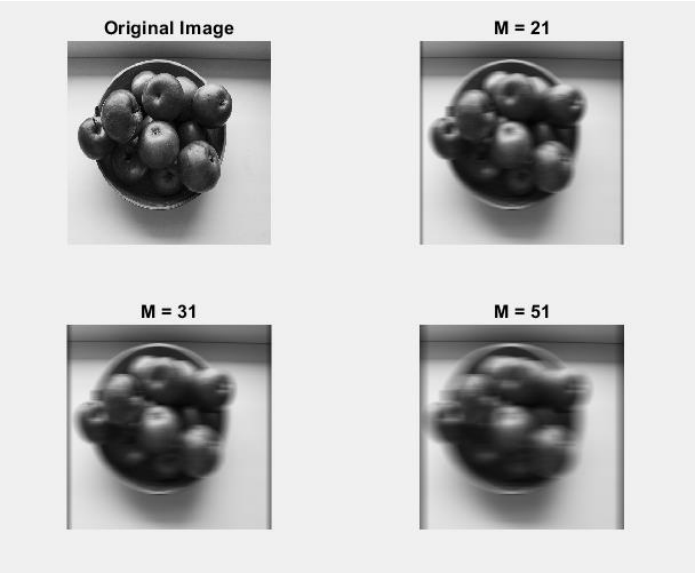

Figure 1: Averaging the image with M-point averaging filter where M = 21, 31, and 51

(i) The visual effect created by the filter is a blurring effect. This effect is due to the averaging of pixel values with their neighbors, which reduces the high-frequency content or the details in the image. To put it in a real-life example, the visual effect of the filter on the image is similar to gently smudging the surface of a painting with a piece of tissue to blend the colors in the image to reduce sharp lines and create a smooth transition between shades. As in the above filter, when you apply harsher smudging, the details of the painting will be less visible. However, this effect will be solely in the horizontal direction, as no vertical averaging or smudging occurs, leaving the vertical details relatively intact and creating a directional blurring effect on only the horizontal axis.

(ii) As the value of M increases, the details in the image become less distinguishable. This is because a larger M means that more neighboring pixels are averaged together, which further smooths out the variations and reduces the sharpness of the image. Also, the blackish area increases as the value of M increases, since more pixels become affected by the zero values lying outside of the image. One final thing to notice is that there is no blackish area in the vertical direction since we apply the averaging only in the horizontal direction.

(iii) The visual effect in the dimension p (vertical) versus the dimension q (horizontal) is likely to be different because of only applying the filter along one dimension (q). This means that the blurring effect is primarily horizontal, and vertical details may be preserved to a greater extent than horizontal details. The filter applied to the images leads to anisotropic blurring whereas, applying the filter in both dimensions would lead to isotropic blurring [1].

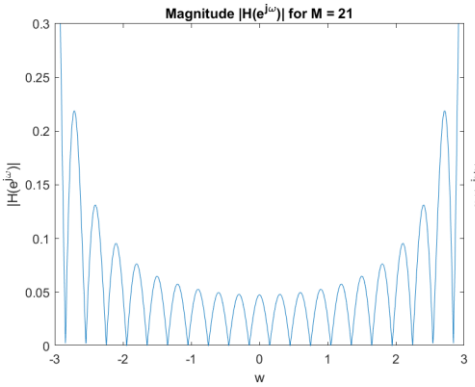

Figure 2: Magnitude of Freq. Response for M = 21

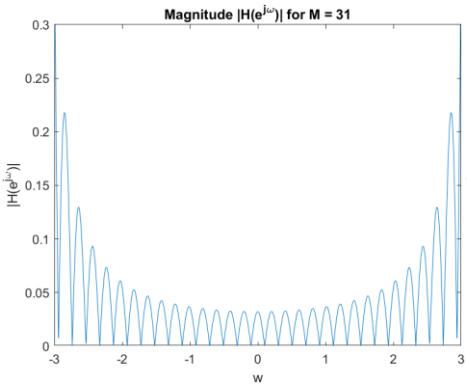

Figure 3: Magnitude of Freq. Response for M = 31

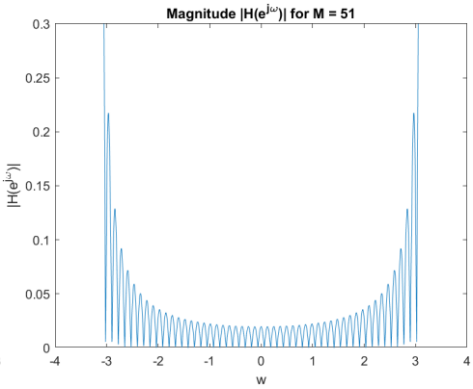

Figure 4: Magnitude of Freq. Response for M = 51

(iv) The size of the frequency response seen in the photos indicates the filter's capacity to suppress different frequency components. Low-frequency elements are associated with smooth surfaces, while high-frequency elements are associated with edges and features (Black is denoted by 1 and white by 0, which means blacks correspond to the high-frequency elements). The averaging filter acts as a low-pass filter, which means it is likely to preserve the low-frequencies. Peaks of the response curves indicate maintained frequencies and dips indicate suppressed frequencies. As M grows, the period of the freq. response plot is decreasing. Therefore, the filter filters the pixels more often. Also, more frequencies, especially the higher frequencies, are muted when M values are larger, which essentially leads to more smoothing.

(v) When approaching the borders of the image, it is expected to observe distortions due to the filter overshooting, which is commonly called as "ringing distortion". As discussed earlier, the reason behind this is that the filter is averaging across the boundary where there is no data. This boundary is assumed to be zero-padded. The degree of ringing effect would be more noticeable with larger M values. This is because the averaging is done over a larger number of pixels. Thus, the transition from the image to the padded zeros is averaged over a wider area, which may lead to more visible distortions at the edges.

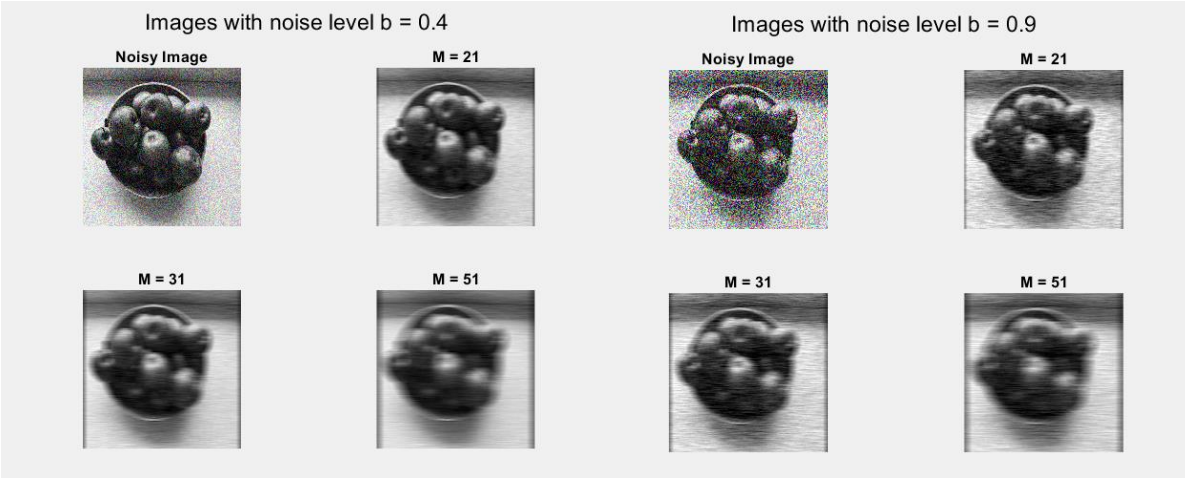

Figure 5: Averaging the noised image with M-point averaging filter where M = 21, 31, and 51; noise level b = 0.4, 0.9

(vi) Yes, averaging does help to reduce the noise. This is evident from the cleaner appearance of the images after the averaging filter is applied. As M increases, the noise reduction becomes more effective because the filter averages over a larger set of pixels, which further smooths out the random variations caused by noise.

(vii) The undesirable side effect of using the averaging filter to eliminate noise is the loss of detail in the image. As M increases, the images become blurrier, and details are lost, which leads to a less sharp image.

(viii) In terms of striking the balance of noise reduction and detail preservation, the best choice of M is when it equals 31. It reduces noise significantly while still retaining more detail compared to M = 51.

2) First Differencer Filter

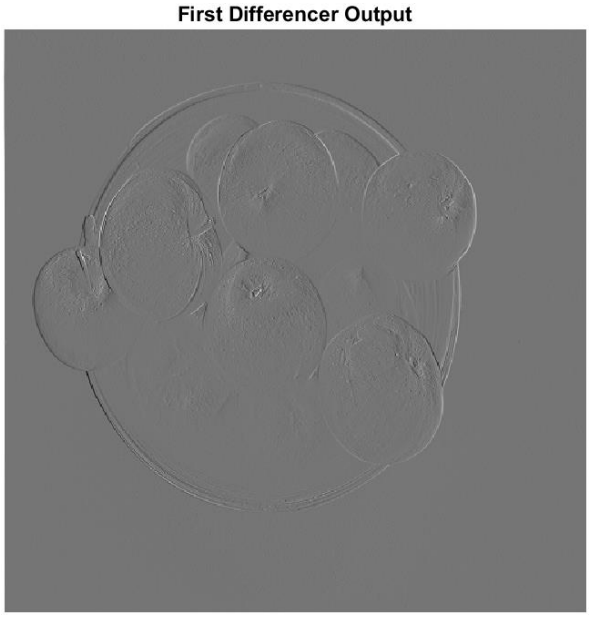

Figure 6: The output image of applying the first differencer filter to the input image.

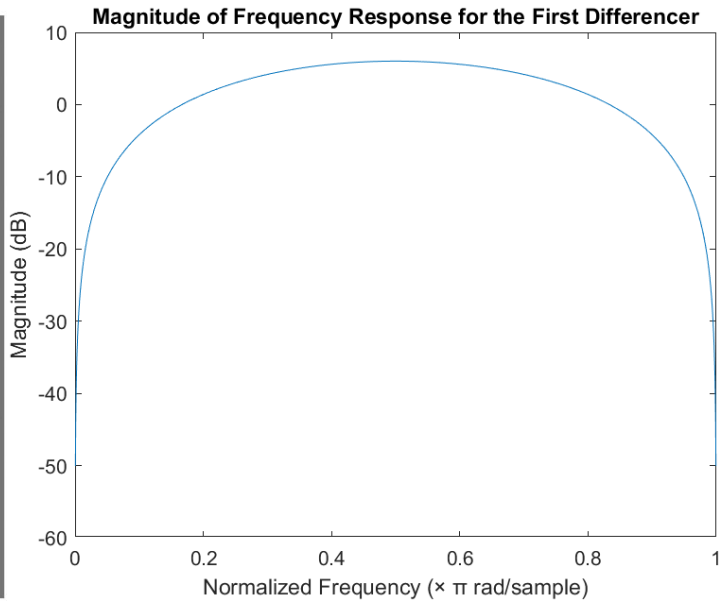

Figure 7: Magnitude of Frequency Response of the first differencer vs. Normalized Frequency Graph

(i) The visual effect created by the first differencer filter is highlighting edges and transitions within the image. It emphasizes the boundaries between different intensity regions, producing an image with boundaries that are computed by the sharp differences of the neighboring pixels. Also, since the background consists of only one color, the filter successfully removes the background.

(ii) The visual effect complies with the theoretical expectation. The first differencer filter acts as a high-pass filter, emphasizing the high-frequency components associated with rapid intensity changes, which, in images, corresponds to the edges.

(iii) Since the filter is applied only in the horizontal dimension, it emphasizes horizontal changes more than vertical ones. For example, we do not see an outline caused by the table to

shadow transition at the top part of the original image since it does not have horizontal transitions. However, if the vertical first differencer had been used, the difference should have been seen after applying the filter.

(iv) The magnitude of the frequency response indicates that this filter allows high-frequency components to pass while eliminating low-frequency components. In the context of an image, this means that sharp details and edges are preserved or even enhanced by the filter, while smooth gradients and areas of uniform intensity are diminished.

Matlab code:

```
I = imread('Apples.bmp');

Igray = mat2gray(I, [0 255]);

[rows, cols] = size(Igray);

outputImage_M21 = oneDimAveragingFilter(Igray, 21);

outputImage_M31 = oneDimAveragingFilter(Igray, 31);

outputImage_M51 = oneDimAveragingFilter(Igray, 51);

figure;
subplot(2, 2, 1), imshow(Igray), title('Original Image');
subplot(2, 2, 2), imshow(outputImage_M21), title('M = 21');
subplot(2, 2, 3), imshow(outputImage_M31), title('M = 31');
subplot(2, 2, 4), imshow(outputImage_M51), title('M = 51');

% filter lengths
Ms = [21, 31, 51];

for i = 1:length(Ms)
    M = Ms(i); % Current M value
    b = ones(1, M) / M; % Averaging filter coefficients
    a = 1; % FIR filter denominator (single pole at zero)

    % frequency response calculation
    [H, omega] = freqz(b, a, 'whole', 2000);

    figure;
    plot(omega-pi, abs(H));
    title(['Magnitude |H(e^{j\omega})| for M = ', num2str(M)]);
    xlabel('w');
    ylabel('|H(e^{j\omega})|');
    ylim([0, 0.3]);
end

b_values = [0.4, 0.9];  % The b values for noise coeffiecients

for b = b_values
    % Noise addition
    noise = (rand(size(Igray)) - 0.5) * b;
```

```
    noisyImage = Igray + noise;

    outputImage_M21_noisy = oneDimAveragingFilter(noisyImage, 21);
    outputImage_M31_noisy = oneDimAveragingFilter(noisyImage, 31);
    outputImage_M51_noisy = oneDimAveragingFilter(noisyImage, 51);

    figure;
    sgtitle(['Images with noise level b = ', num2str(b)]);
    subplot(2, 2, 1), imshow(noisyImage), title('Noisy Image');
    subplot(2, 2, 2), imshow(outputImage_M21_noisy), title('M = 21');
    subplot(2, 2, 3), imshow(outputImage_M31_noisy), title('M = 31');
    subplot(2, 2, 4), imshow(outputImage_M51_noisy), title('M = 51');
end

figure;
outputImage_diff = firstDifferencer(Igray);
imshow(outputImage_diff, []);
title('First Differencer Output');

b = [1, -1];  % first differencer filter coefficients
a = 1;        % Denominator coefficient (single pole at zero)

% Frequency response
[H_diff, omega] = freqz(b, a, 'whole', 2000);

% Normalized frequency values
omega_normalized = omega / (2 * pi);
figure;

plot(omega_normalized, 20 * log10(abs(H_diff)));
title('Magnitude of Frequency Response for the First Differencer');
xlabel('Normalized Frequency (× π rad/sample)');
ylabel('Magnitude (dB)');


function outputImage = firstDifferencer(inputImage)
    [rows, cols, ~] = size(inputImage);
    outputImage = zeros(rows, cols);
    outputImage(:, 1) = inputImage(:, 1); % first column will not change

    for i = 1:rows
        for j = 2:cols
            outputImage(i, j) = inputImage(i, j) - inputImage(i, j - 1);
        end
    end
end

function outputImage = oneDimAveragingFilter(inputImage, M)
    [rows, cols, ~] = size(inputImage);

    outputImage = zeros(rows, cols);

    fprintf('Rows: %d, Cols: %d\n', rows, cols);

    for i = 1:rows
```

```
        for j = 1:cols
            % The left and right indices for averaging
            leftIndex = max(1, j - (M - 1) / 2);
            rightIndex = min(cols, j + (M - 1) / 2);

            % Column vector extraction (I do not add the 0 values explicitly but
later I divide by M)
            columnVector = inputImage(i, leftIndex:rightIndex);

            % Average calculation
            outputImage(i, j) = sum(columnVector) / M;
        end
    end
end
```

References:

[1] “Anisotropic filtering,” DevX, https://www.devx.com/terms/anisotropic-filtering/ (accessed Dec. 18, 2023).