CSE 3113 / CSE 3214

INTRODUCTION TO DIGITAL IMAGE PROCESSING

SPRING 2024

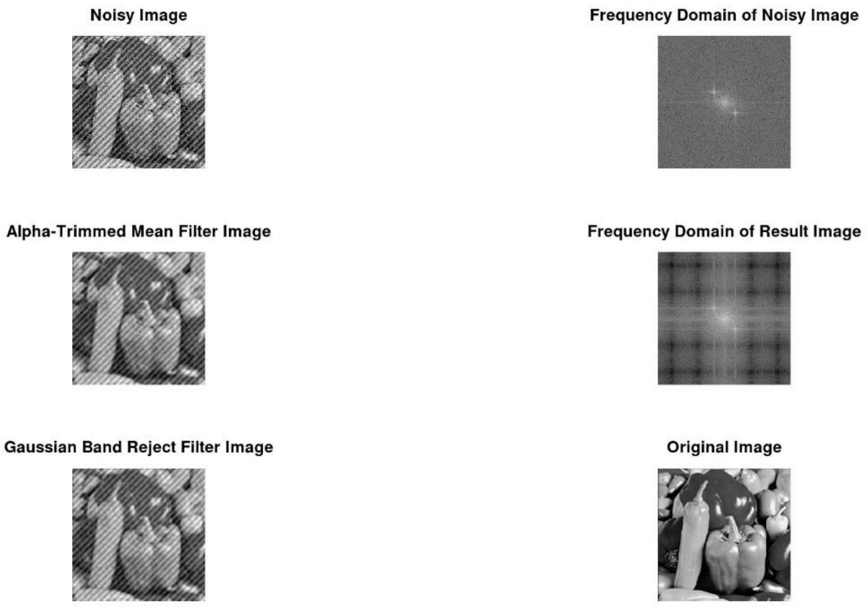
Homework 3 Report

İkram Celal KESKİN – 200316059
Image-100

Submission Date: 17 May 2024

Programming Language	☐ *Python* ☐ *Matlab* [✓ *Octave*]
Programming Environment	GNU Octave, version 8.4.0
Reflections	I tried to remove the periodric noise, but I failed

Results & Discussion


Noisy Image
Frequency Domain of Noisy Image
Alpha-Trimmed Mean Filter Image
Frequency Domain of Result Image
Gaussian Band Reject Filter Image
Original Image

- The major noise types apparent in the input image are salt-and-pepper noise and periodric noise. Salt-and-pepper noise appears as random white and black pixels scattered throughout the image. This type of noise is typically caused by sharp, sudden disturbances in the image signal.

- To remove salt-and-pepper noise, I used the alpha-trimmed mean filter. This filter is effective because it can remove extreme pixel values (outliers) from the computation of the mean value, which are characteristic of salt-and-pepper noise. By trimming the alpha percentage of the smallest and largest pixel values, the filter can average the remaining values, thus reducing the impact of the noise.

- For reducing periodric noise, I applied a Gaussian band reject filter. This technique is effective because it targets specific frequency bands where the noise is prominent. The Gaussian band reject filter attenuates these frequencies while preserving the important features of the image.

- **Best Parameters for Techniques**

  a. Alpha-Trimmed Mean Filter:
    - Window Size: 5x5
    - Alpha: 0.075
    - These parameters provided a good balance between noise removal and edge preservation, effectively reducing the salt and pepper noise without introducing significant blurring.

  b. Gaussian Band Reject Filter:
    - Frequency Center (f_center): 20
    - Rate (rate): 100
    - These values were effective in targeting the specific frequencies associated with the periodic noise, resulting in a cleaner image while maintaining important image details.

Source Code

```
imageNoisy = imread('./image100.png');

imageOriginal = imread('./0.tif');


pkg image load;


function output_img = alpha_trimmed_mean_filter(input_img, window_size, alpha)

   [m, n] = size(input_img);

   output_img = zeros(m, n);


   pad_size = floor(window_size / 2);

   padded_img = padarray(input_img, [pad_size, pad_size], 'replicate');


   for i = 1:m

      for j = 1:n

         window = padded_img(i:i+window_size-1, j:j+window_size-1);

         % Get the sorted values of the window
```

```
            sorted_window = sort(window(:));
            % Calculate the number of pixels to exclude on each side
            exclude_count = floor(alpha * window_size^2 / 2);
            % Remove the alpha percentage of pixels from both ends
            trimmed_values = sorted_window(exclude_count+1:end-exclude_count);
            % Compute the alpha-trimmed mean
            output_img(i, j) = mean(trimmed_values);
        end
    end
end
%-------------------------------------------------------------------------
function output_img = gaussian_band_reject_filter(input_img, f_center, rate)

    F = fft2(input_img);
    F_shifted = fftshift(F);

    [rows, cols] = size(input_img);
    u = 0:(rows-1);
    v = 0:(cols-1);
   idx = find(u > rows/2);
    u(idx) = u(idx) - rows;
    idy = find(v > cols/2);
    v(idy) = v(idy) - cols;
    [V, U] = meshgrid(v, u);
    D = sqrt(U.^2 + V.^2);

    H = 1 - exp(-((D.^2 - f_center^2).^2) / (2*(rate^2)*(f_center^2)));
    G = H .* F_shifted;

  G_shifted = ifftshift(G);
  img_filtered = ifft2(G_shifted);
  output_img = abs(img_filtered);
end

figure('Position', [100, 100, 1200, 800]);
%--------------------------------------------------------------------------
subplot(3,2,1);
imshow(imageNoisy);
title('Noisy Image');
set(gca, 'FontSize', 12);
%--------------------------------------------------------------------------
fft_img = fft2(imageNoisy);
fft_shifted = fftshift(fft_img);
fft_result_image = mat2gray(abs(log(1 + fft_shifted)));

subplot(3,2,2);
imshow(fft_result_image, []);
title('Frequency Domain of Noisy Image');
set(gca, 'FontSize', 12);
%--------------------------------------------------------------------------
result_img= alpha_trimmed_mean_filter(imageNoisy,5,0.075);
subplot(3,2,3);
imshow(result_img,[]);
```

```
title('Alpha-Trimmed Mean Filter Image');

set(gca, 'FontSize', 12);

%--------------------------------------------------------------------------

fft_img = fft2(result_img);

fft_shifted = fftshift(fft_img);

fft_result_image = mat2gray(abs(log(1 + fft_shifted)));


subplot(3,2,4);

imshow(fft_result_image, []);

title('Frequency Domain of Result Image');

set(gca, 'FontSize', 12);

%--------------------------------------------------------------------------

resultGaussian_img= gaussian_band_reject_filter(result_img,20,100);

subplot(3,2,5);

imshow(resultGaussian_img,[]);

title('Gaussian Band Reject Filter Image');

set(gca, 'FontSize', 12);

%--------------------------------------------------------------------------

subplot(3,2,6);

imshow(imageOriginal, []);

title('Original Image');

set(gca, 'FontSize', 12);
```