CSE 3113 / CSE 3214

INTRODUCTION TO DIGITAL IMAGE PROCESSING

SPRING 2024

Homework 3 Report

Bilal AYAKDAŞ – 220316002

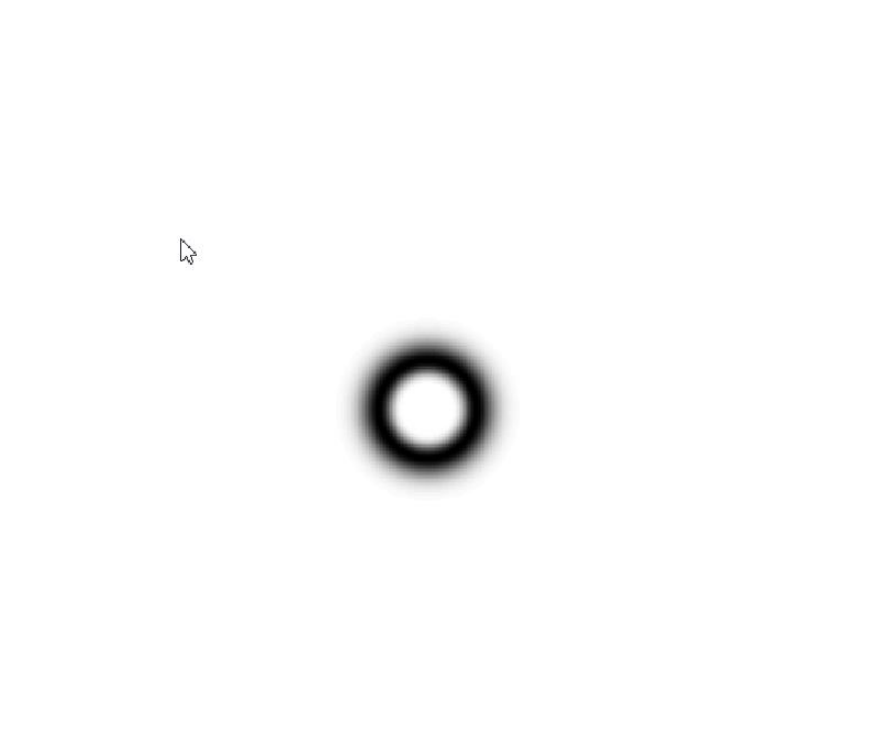
Submission Date: 15 May 2024

Programming Language	☑ *Python* ☐ *Matlab* ☑ *Octave*
Programming Environment	*I used version 8.4.0 of the octave.*
Reflections	*I learned how to remove salt and pepper noise from the image and how to remove periodic noise from the image with Fourier transform. but I could not remove the noise completely.*

Results & Discussion

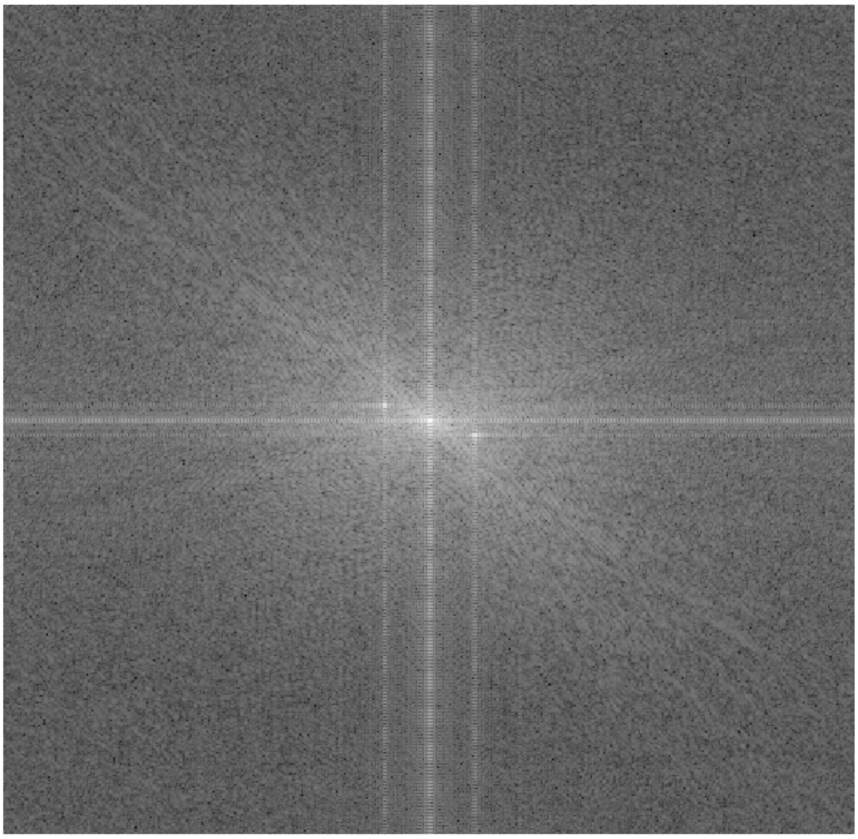
1. *Paste the output figures that you generated.*

Filter in the frequency domain

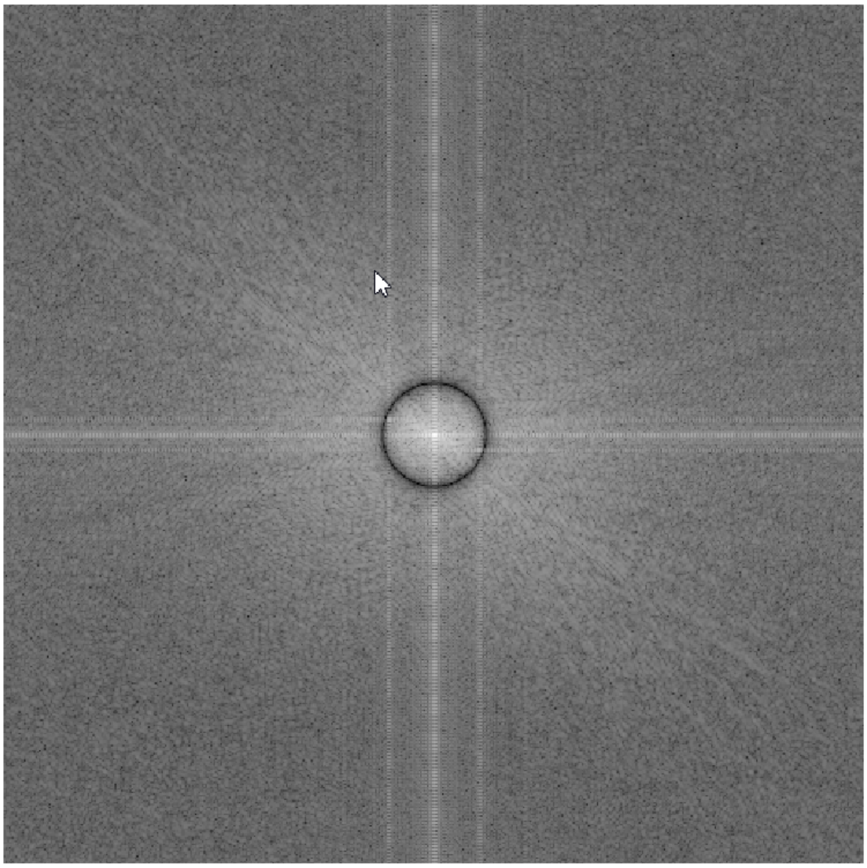


Filtered image

Fourier spectrum of the original image



Fourier spectrum of the filtered image



2. *Make a discussion about your results around the following questions.*
    a. *What are the major noise types apparent in your input image?*
        There is salt and paper and periodic noise in the input image

    b. *Which techniques have you used to remove each type of noise? Why?*
        I used a median filter to remove salt and pepper noise.
    Because salt and pepper noise consist of extreme values and when I used the median filter, the extreme values disappeared.
        I used a notch filter to remove periodic noise. Because there was periodic distortion in the frequency domain in the x and y axis in the picture, I was able to eliminate this with a notch filter.

    c. *What are the best parameters of those techniques, for your input?*

    D0 = 0.06*PQ(1);
    H = bandreject('gaussian', PQ(1), PQ(2), D0, 20);

    I used 6 percent of the image as the diameter of the circle . 20 as the width of the filter.

Source Code

dtuv.m

```
function [U, V] = dftuv(M, N)

%DFTUV Computes meshgrid frequency matrices.

%[U, V] = DFTUV(M, N) computes meshgrid frequency matrices U
and

%V. U and V are useful for computing frequency-domain filter

%   functions that can be used with DFTFILT.  U and V are both
M-by-N.
```

```
% Set up range of variables.
u = 0:(M-1);
v = 0:(N-1);
% Compute the indices for use in meshgrid
idx = find(u > M/2);
u(idx) = u(idx) - M;
idy = find(v > N/2);
v(idy) = v(idy) - N;
% Compute the meshgrid arrays
[V, U] = meshgrid(v, u);
```

paddedsize.m

```
function PQ = paddedsize(AB, CD, PARAM)
%PADDEDSIZE  Computes  padded  sizes  useful  for  FFT-based
filtering.
%PQ = PADDEDSIZE(AB), where AB is a two-element size vector,
%computes the two-element size vector PQ = 2*AB.
%
%PQ = PADDEDSIZE(AB, 'PWR2') computes the vector PQ such that
%PQ(1) = PQ(2) = 2^nextpow2(2*m), where m is MAX(AB).
%
%PQ = PADDEDSIZE(AB, CD), where AB and CD are two-element size
%vectors, computes the two-element size vector PQ.The elements
%of PQ are the smallest even integers greater than or equal to
%   AB + CD -1.
%
%PQ  =  PADDEDSIZE(AB,  CD,  'PWR2')  computes  the  vector  PQ  such
that
%   PQ(1) = PQ(2) = 2^nextpow2(2*m), where m is MAX([AB CD]).
```

```
if nargin == 1
   PQ = 2*AB;
elseif nargin == 2 & ~ischar(CD)
   PQ = AB + CD - 1;
   PQ = 2 * ceil(PQ / 2);
elseif nargin == 2
   m = max(AB); % Maximum dimension.

   % Find power-of-2 at least twice m.
   P = 2^nextpow2(2*m);
   PQ = [P, P];
elseif nargin == 3
   m = max([AB CD]); %Maximum dimension.
   P = 2^nextpow2(2*m);
   PQ = [P, P];
else
   error('Wrong number of inputs.')
end
```

bandreject.m

```
function H = bandreject(type, M, N, D0, W, n)
%bandreject Computes frequency domain bandreject filters
%    H = bandreject(TYPE, M, N, D0, n) creates the transfer function of
%   a bandreject filter, H, of the specified TYPE and size (M-by-N).  To
```

```
%    view the filter as an image or mesh plot, it should be
centered
%   using H = fftshift(H).
%
%   Valid values for TYPE, D0, and n are:
%
%   W :    is the width of the filter
%
%   'ideal'     Ideal lowpass filter with cutoff frequency D0.
n need
%              not be supplied.  D0 must be positive
%
%'btw' Butterworth lowpass filter of order n, and cutoff D0.
%The default value for n is 1.0.  D0 must be positive.
%
%'gaussian'Gaussian   lowpass   filter   with   cutoff(standard
deviation)
% D0.  n need not be supplied.  D0 must be positive.
% Use function dftuv to set up the meshgrid arrays needed for
% computing the required distances.
[U, V] = dftuv(M, N);


% Compute the distances D(U, V).
D = sqrt(U.^2 + V.^2);
% Begin fiter computations.
switch type
case 'ideal'
   H = 1-double(D >= (D0-W/2)).*double(D <= (D0+W/2));
```

```
case 'btw'
   if nargin == 5
      n = 1;
   end
   H = 1./(1 + (D*W./(D.^2-D0*D0)).^(2*n));
case 'gaussian'
   H = 1-exp(-((D.^2 - D0*D0)./(D*W)).^2);
otherwise
   error('Unknown filter type.')
end
```

ban IDIP_HW3_NoiseReduction.m

```
clearvars, close all, clc;
I1=imread('image116.png');
imshow(I1), title('Original image')
I = medfilt2 (I1)
imshow(I), title('Median Filtered İmage')
%Determine good padding for Fourier transform
PQ = paddedsize(size(I));
%Create a bandreject filter
D0 = 0.06*PQ(1);
H = bandreject('gaussian', PQ(1), PQ(2), D0, 20);
figure,imshow(fftshift(H)),  title('Filter  in  the  frequency
domain');
% Calculate the discrete Fourier transform of the image
F=fft2(double(I),size(H,1),size(H,2));
% Apply the filter to the Fourier spectrum of the image
BRFS_I = H.*F;
```

```
% convert the result to the spatial domain.

BRF_I=real(ifft2(BRFS_I));


% Crop the image to undo padding

BRF_I=BRF_I(1:size(I,1), 1:size(I,2));


%Display the filtered image

figure, imshow(BRF_I,[]), title('Filtered image')


% Display the Fourier Spectrum

% Move the origin of the transform to the center of the
frequency rectangle.

Fc=fftshift(F);

Fcf=fftshift(BRFS_I);

% use abs to compute the magnitude and use log to brighten
display

S1=log(1+abs(Fc));

S2=log(1+abs(Fcf));


figure,imshow(S1,[]), title('Fourier spectrum of the original
image');

figure, imshow(S2,[]), title('Fourier spectrum of the filtered
image');
```