Image Processing
lab 2

Klaas Kliffen Jan Kramer

December 3, 2015

Exercise 1 – Fourier spectrum

a. The functions build in functions fft2 and fftshift are used to create the fourier spectrum image centered around the DC component of the fourier transform. Some extra code is added for calculating the average of the image, which will be explained in more detail in part c of this exercise. To be able to view the spectrum, the values need to be scaled. So the log is taken of each value to increase the constrast.

```
% Read the image
x = imread('../images/characters.tif');
% Get image width and heigt
[width, height] = size(x);
% Perform the Fourier transform
spectrum = fftshift(fft2(x));
% Calculate the avarge of the image
% It can be found by taking the dc component (center of the image)
% And dividing it by the number of pixels
avg_fourier = abs(spectrum(width/2+1,height/2+1))/(width*height)
avg_mean = mean(mean(x))

% calculate the magnitude
spectrum = abs(spectrum);

% take the log value for better scaling in octave
logspectrum = log2(spectrum);
% Take note of max and min of the spectrum for image scaling
maxs = max(max(logspectrum));
mins = min(min(logspectrum));
% Scale the image for output to file
spectrumimg = uint8(floor((logspectrum - mins) / (maxs-mins) * 256));
imwrite(spectrumimg, 'spectrum.png');
% Show the log image as a figure
figure, imshow(logspectrum,[]), colormap gray
```

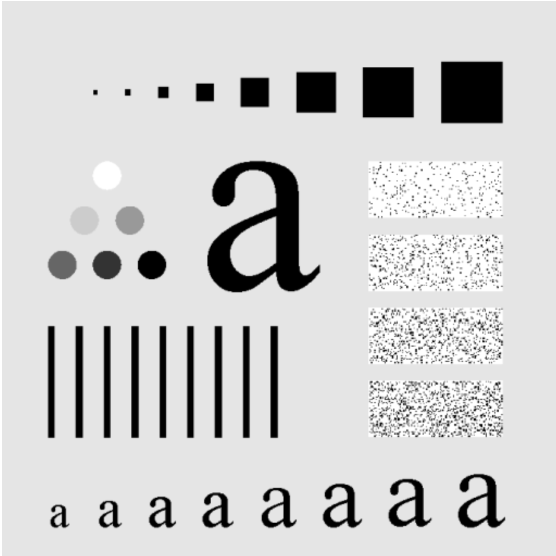

Figure 1: Original image

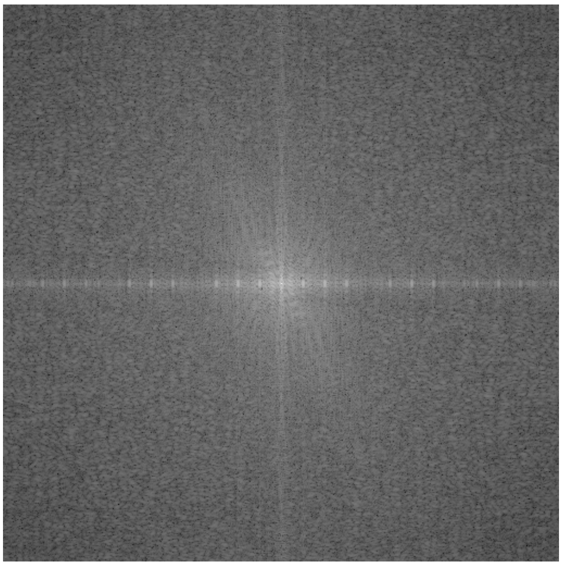

Figure 2: Fourier spectrum of figure 1

b.

c. The maginitude of the center of the image is the DC component. For images, this is the sum of the grey levels of all pixels. Dividing this by the number of pixels gives the average grey level. In this case: 207.31 for figure 1. This value is equal to the result achieved by using `mean` two times.

Exercise 2 – Highpass filtering in the frequency domain

a.

b.

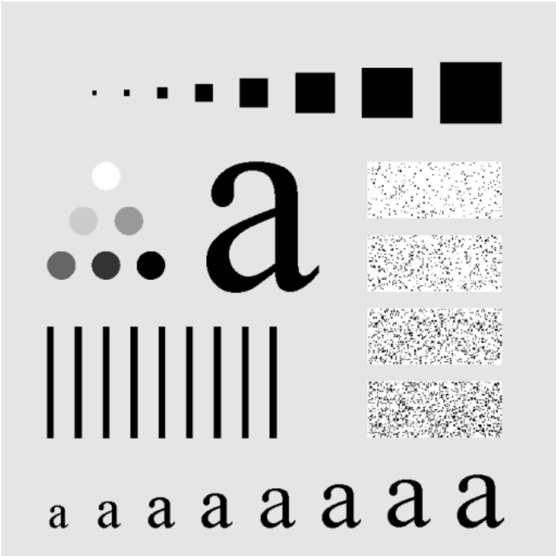

Figure 3: Original image

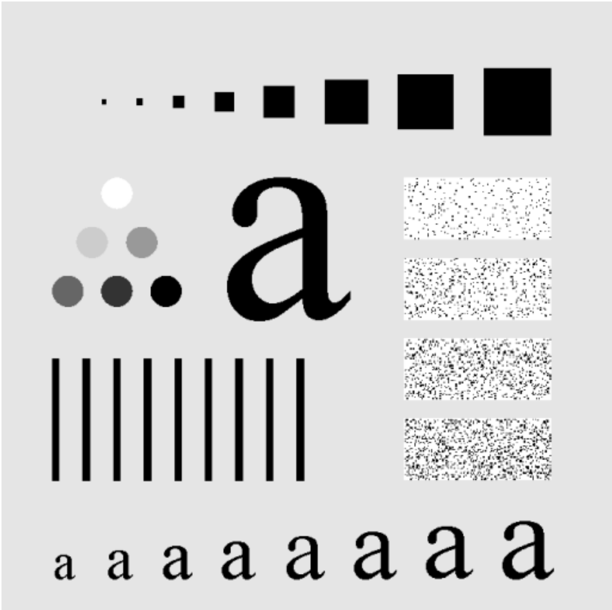

Figure 4: Gausian filtering on figure 3

c.

Exercise 3 – Median filtering

a.

```
% Median filter width a 2k+1 x 2k+1 window
function [out] = IPmedian(img,k)
% get the image size
[width, height] = size(img)
%create the output image
dest = uint8(zeros(size(img)));

% loop over all pixels
for x = 1 : width
  for y = 1 : height
    % determine the edge of the window
    % the size of the window shrinks at the boundaries of the image
    startx = max(x-k,1);
    starty = max(y-k,1);
    endx = min(x+k,width);
    endy = min(y+k,height);
    % get the submatrix
    submat = img(startx:endx,starty:endy);
    [w,h] = size(submat);
    % convert it to a vector to be able to calculate the median
    submat = reshape(submat, 1, w*h);
    % take the median and store it
    dest(x,y) = median(submat);
  end
end
% output the image
out = dest;
```

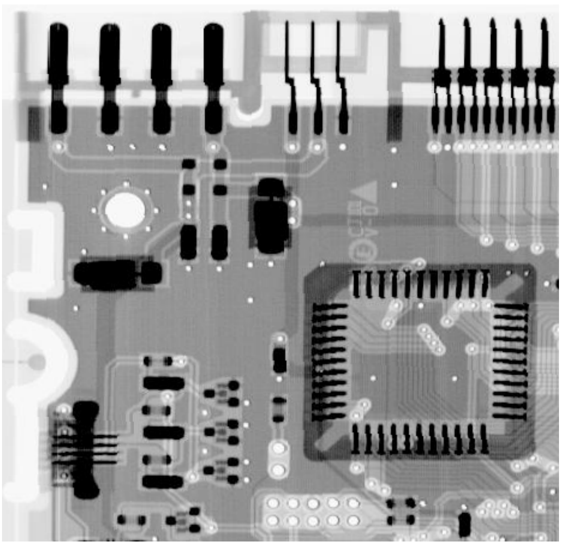

Figure 5: Original image of the circuitboard

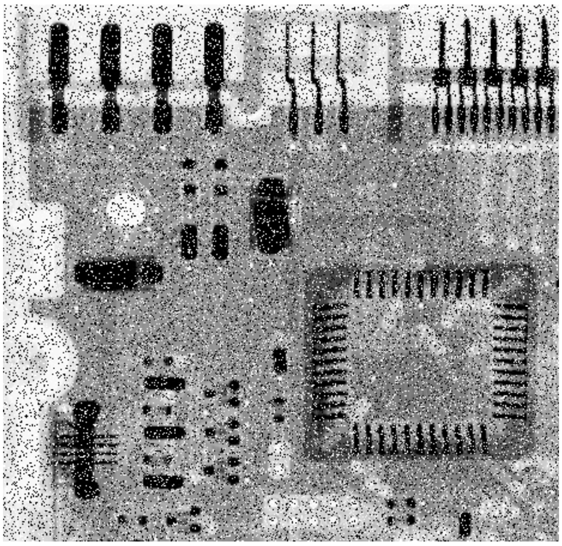

Figure 6: Figure 5 with salt & pepper noise with $Pa = Pb = 0.2$

b.

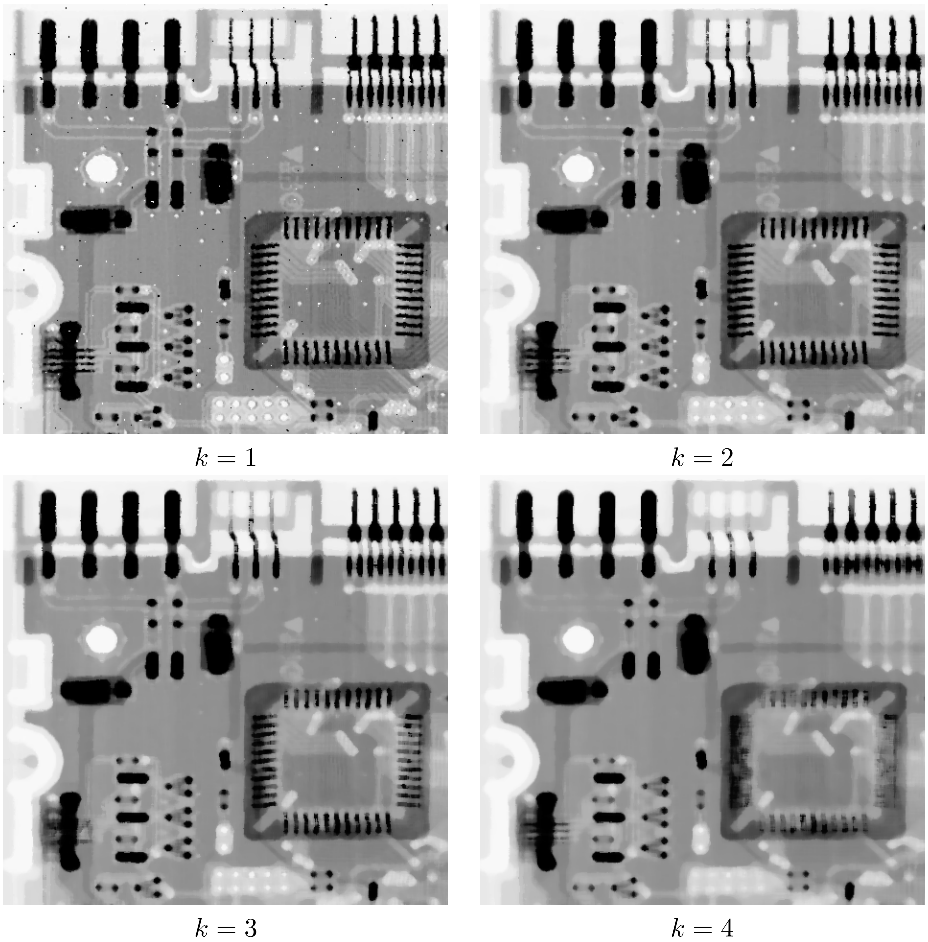

Figure 7: Using median filtering with different window values k on figure 6

c.

Task distribution

ex1	design	implementation	answers questions	writing report
Klaas	50%	100%	50%	50%
Jan	50%	0%	50%	50%

ex2	design	implementation	answers questions	writing report
Klaas	50%	0%	50%	50%
Jan	50%	100%	50%	50%

ex3	design	implementation	answers questions	writing report
Klaas	50%	100%	50%	50%
Jan	50%	0%	50%	50%