Module assignment front page

Course code and name: FYS-2010 Digital Image Processing

Candidate number: 20

Date and year: 18.03.2016

Contains 11 numbered sheets, front page included.

FACULTY OF SCIENCE AND TECHNOLOGY

Contents

Problem 1. Histogram Equalization

A)

A histogram provides information regarding the intensity levels of the pixels in an image, using this we can discover some of its characteristics. The graph shows the number of occurences of each intensity level and it is often normalized to a probablility estimate by dividing with the total number of pixels. By looking how the graph is concentrated we can see features of the image, such as darkness/brightness or low/high contrast. We can also enhance the image to our needs by manipulating its histogram. [1]

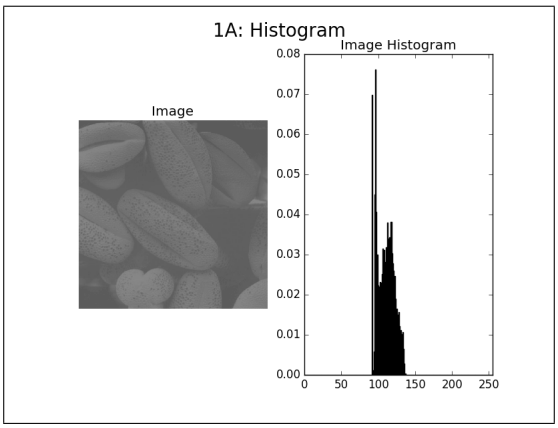

Figure 1: Result 1A

Looking at the results we see that the histogram is concentrated in the middle of the intensity scale and it is very narrow. This suggests a low contrast image with a lot of monotonous gray color, since the image is grayscale.

B)

It is possible to modify the intensities of an image to enhance it's contrast using the histogram, this procedure is called histogram equalization. We get a contrast enhancement by mapping the intensity values of pixels in the input image to an output image such that the intensity distribution is spread to a wider range. This is done by the mapping equation [2]

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) \text{ for } k = 0, 1, 2, ..., L-1$$

Where each pixel with intensity level r_j in the input image is mapped to a corresponging pixel with intensity level s_k. $p_r(r_j)$ is the probability of occurrence of intensity level r_j and L is the number of intensity levels.

In other words, we are linearizing the cumulative distribution function.

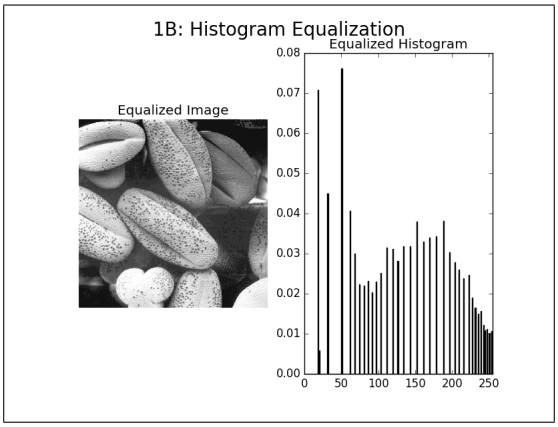

Figure 2: Result 1B

We can see that the equalized histogram has a much wider spread than the original one, which results in a high contrast image with much more detail. The high values on the lower end of the scale are more spread compared to the lower values on the high end of the scale, this makes sense since we have a uniform probability density function.

Problem 2. Spatial Filtering

A)

```
[[0 0 0 0 0]
 [0 1 2 3 0]
 [0 4 5 6 0]
 [0 7 8 9 0]
 [0 0 0 0 0]]
```

Figure 3: Result 2A

Convoluting a filter with a unit impulse image gives a copy of the filter at the position of the impulse. To convolute we pad the image, rotate the filter 180 degrees, move it over the image and compute the sum of products on each position. If we dont rotate the filter, we would get the correlation. (rotated filter at the position of the impulse) [3]

B)

Laplacian filters are very commonly used when sharpening images. When convoluting a laplacian filter with an image, we get a laplacian image $\bigtriangledown^2 f(x, y)$ that highlights the features where intensities are discontinuous, while downplaying the areas that are continuous. Adding the laplacian image to the original image $f(x, y)$ gives us an output $g(x, y)$ where edges and details of the original image are sharpened while the background is more or less the same.
The process can be shown by a simple equation [4]
$g(x, y) = f(x, y) + c[\bigtriangledown^2 f(x, y)]$ where the constant $c = -1$ or 1 depending on the filter used.

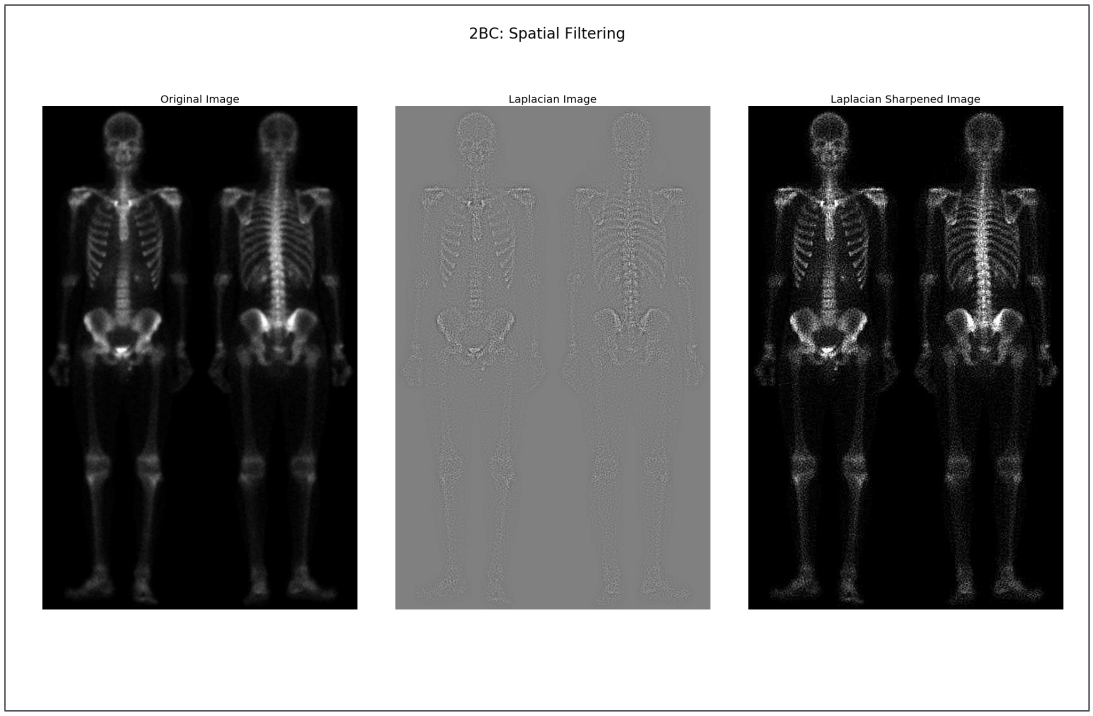

Figure 4: Result 2BC

Problem 3. Denoising

A)

By visually investigating the three images we can see pretty clear what kind of noise has been added, atleast for "P3_fig2.png" and "P3_fig3.png". Salt-and-pepper noise is added to the second image and the third image has periodic noise. The noise in "P3_fig1.png" resembles gaussian noise, but it is hard to visually distinguish from other types of noise. Below we have some histograms and power spectrum to precisely see what kind of noise we are dealing with and its characteristics. The histograms are derived from subimages with reasonably constant background intensity to distinguish the noise [5]. Now we can clearly see that the first image is corrupted by additive gaussian noise because of the shape of the histogram. As expected, the second image is corrupted by salt-and-pepper noise and we can see that the probablilites of the noise are $P_a \approx P_b \approx 0.09$. The third image is corrupted by sinusoidal noise, which is evident from the single conjugate impulse pair in its power spectrum.

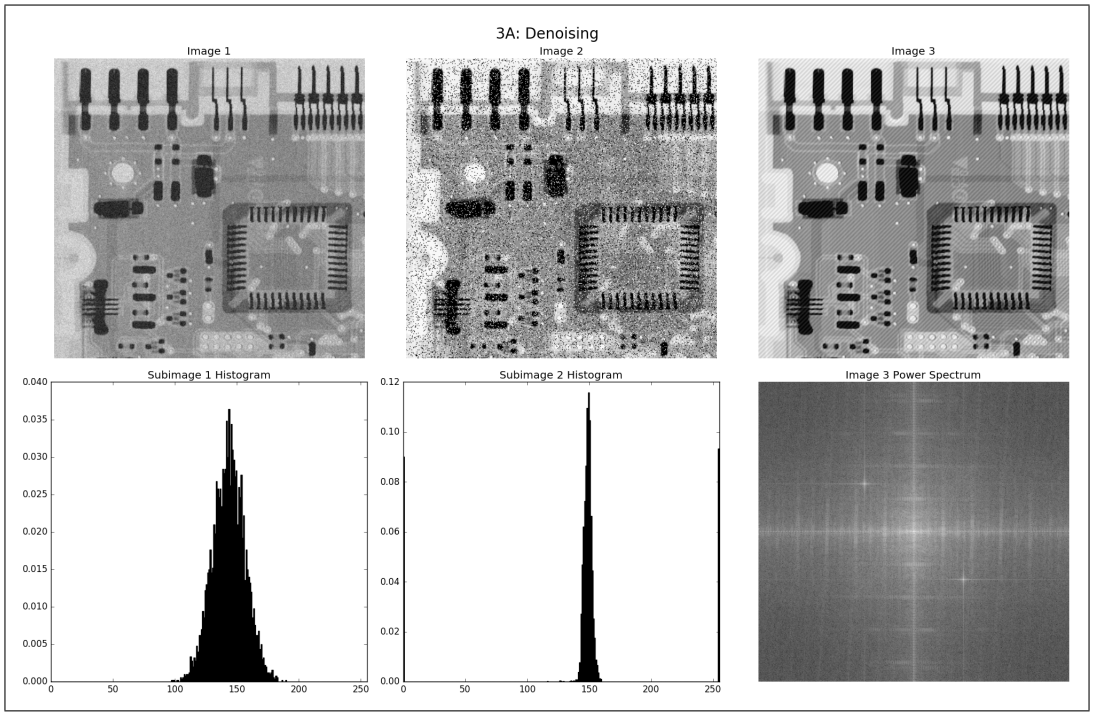

Figure 5: Noise Investigation

B)

For the first image I used an adaptive local noise reduction filter to get rid of the gaussian noise. Adaptive filters change behaviour according to statistical values inside the filter region, which often makes them a better choice than using simpler filters such as arithmetic or geometric mean. There is however a tradeoff in perfomance, which will be discussed later. The adaptive local noise reduction uses the original value $g(x, y)$, the local mean m_L and variance ${\sigma_L}^2$ of the filter region, and a estimate of the overall noise variance ${\sigma_\eta}^2$ of the image.
The output value is based on the following equation [6]

$$\hat{f}(x, y) = g(x, y) - \frac{{\sigma_\eta}^2}{{\sigma_L}^2}[g(x, y) - m_L]$$

From the equation we can see that

- If ${\sigma_\eta}^2$ is zero, the original value is returned since there is no noise.
- If ${\sigma_L}^2$ is high (indicates edges within filter region) relative to ${\sigma_\eta}^2$, the ratio becomes very small and a value close to $g(x, y)$ is returned to preserve edges.
- If the variances are equal (indicates noise within filter region) the ratio is 1 and m_L is returned, to reduce the noise.

The results of our implementation is compared with arithmetic and geometric mean filters.

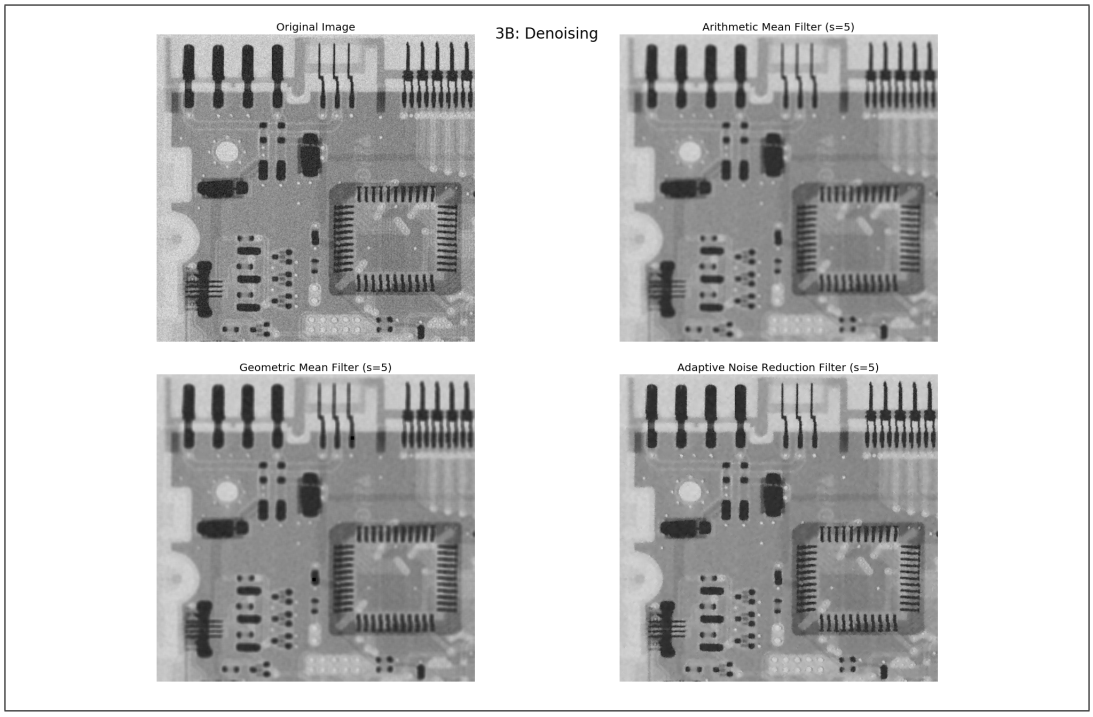

Figure 6: Result using an Adaptive local noise reduction filter, all filters are size 5x5

We see that the filtered images are pretty similar in regards to all the noise being smoothed out, but the image is much sharper using adaptive local noise reduction since the edges are better preserved. Unfortunately this improvement comes with a tradeoff in complexity. The adaptive local noise reduction filter has a significantly slower runtime than arithmetic and geometric mean filter. In this case it spent around 22 seconds to complete while the arithmetic and geometric mean filters spent around 7 seconds. This cost is mostly due to the amount of statistical calculations inside the filter region for each pixel and I imagine the runtime will be even slower with larger images.

C)

To filter a image $f(x, y)$ in the frequency domain you start with padding the image with zeros (usually double the image size), then compute the discrete fourier transform (DFT) $F(u, v)$ of the padded image $f_p(x, y)$. Center the transform by shifting the low frequencies to the center with a shifting function (can also be done in spatial domain). Create a symmetric filter transfer function $H(u, v)$ of the same size as $F(u, v)$ which passes or rejects certain frequencies. Get output by multiplying the tranform with the filter $G(u, v) = F(u, v)H(u, v)$. Shift frequencies back with an inverse shifting function and get output image $g(x, y)$ in spatial domain using inverse discrete fourier transform (IDFT). Finally we get rid of complex values and extract the result image from the padded output $g(x, y)$. [7]

The link between filtering in the spatial and frequency domain is the convolution theorem which states that multiplication in the frequency domain corresponds to circular convolution in the spatial domain and vice versa. [8]

$f(x, y) \star h(x, y) \iff F(u, v)H(u, v)$
$f(x, y)h(x, y) \iff F(u, v) \star H(u, v)$

Padding is important because we will get wraparound error when convolving two periodic functions without it. [8] Wraparound error is when results from one side of the image wraps around and affects the results on the other side. Using zero-padding, this wraparound can take place without affecting the results.

D)

When removing the Gaussian noise in the frequency domain I used a Butterworth lowpass filter. A lowpass filter will pass the lower frequencies and cut off the higher frequencies in the image, which results in a blurring effect since the high frequencies represents intensity discontinuation such as sharp edges and noise. I chose Butterworth because its a middle ground between the Ideal and Gaussian, and it is possible to experiment with parameters to find the best possible result. The filter transfer function is defined as [9]

$$H(u, v) = \frac{1}{1+[D(u,v)/D_0]^{2n}}$$

Where $D(u, v)$ is the euclidean distance from point (u, v) to the center of the frequency rectangle, D_0 is the cutoff frequency (radius where we stop passing frequencies), and n is the order of the filter. This transfer function can provide a smooth transition in blurring with a low order (Gaussian) or a sharp transition in blurring for a high order (Ideal).

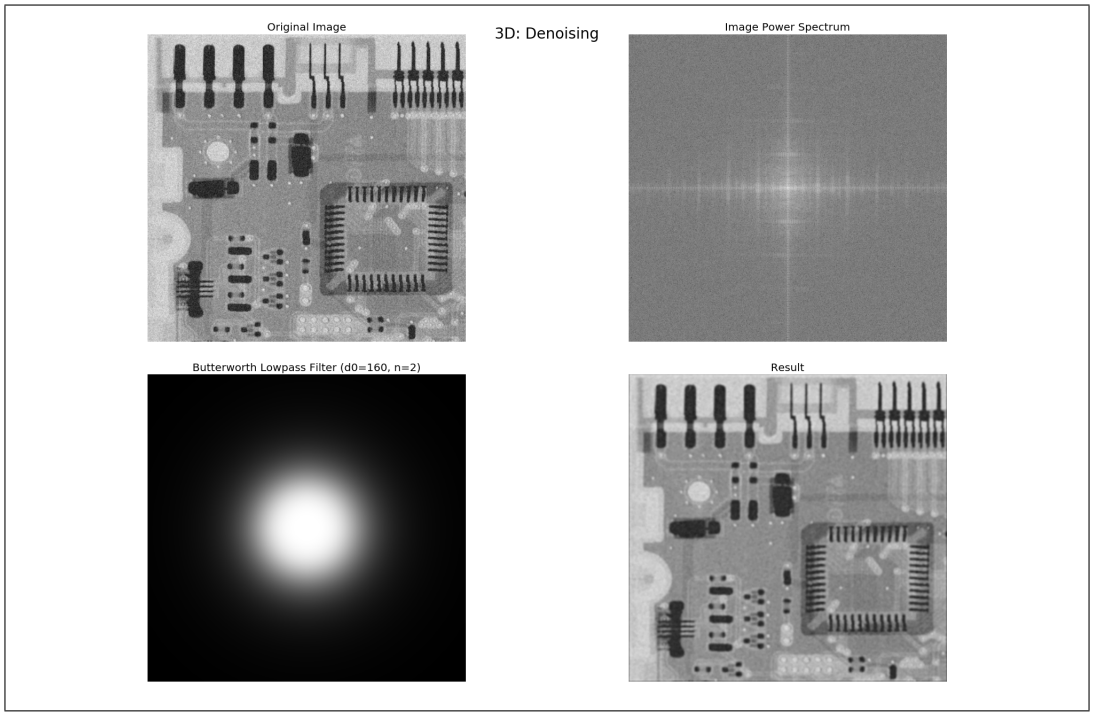

Figure 7: Result using a Butterworth lowpass filter of order 2 and cutoff frequency 160

We can see that there is no clear indication of noise on the power spectrum of the image because it is not periodic. The approach was to experiment with different orders and cutoff frequencies to find a result with minimal noise, but also not blur the image too much. Comparing this result to the spatial domain filtering in 3B, it can be seen as a middle ground between geometric mean and adaptive local noise filtering in regards to picture sharpness. When it comes to noise, it is more like the geometric mean since it seems to be a little coarse in the gray areas.

E)

For the second image i've used an adaptive median filter to remove the salt-and-pepper noise. This filter is very effective for impulse noise since it forces extreme intensity values to be more like their neighbors. It also has other advantages such as preserving details while smoothing nonimpulse noise, which could be useful in images with several different types of noise. This filter method depends on the minimum, maximum, median and center point values inside the filter region, which is dynamic in size.

The algorithm runs as follows [10]

1. If the median is not an impulse, it checks the center point.
    - If the center point is not an impulse, return center point
    - If the center point is an impulse, return median
2. If the median is an impulse, the filter region size is increased and the algorithm is repeated.
3. If the filter region surpassed its maximum size, return median

We see that instead of forcing every point to be the median (like the standard median filter), the algorithm tries to return the center point value if possible, to preserve image details. The results of our implementation is compared with a standard median filter, size 5x5.

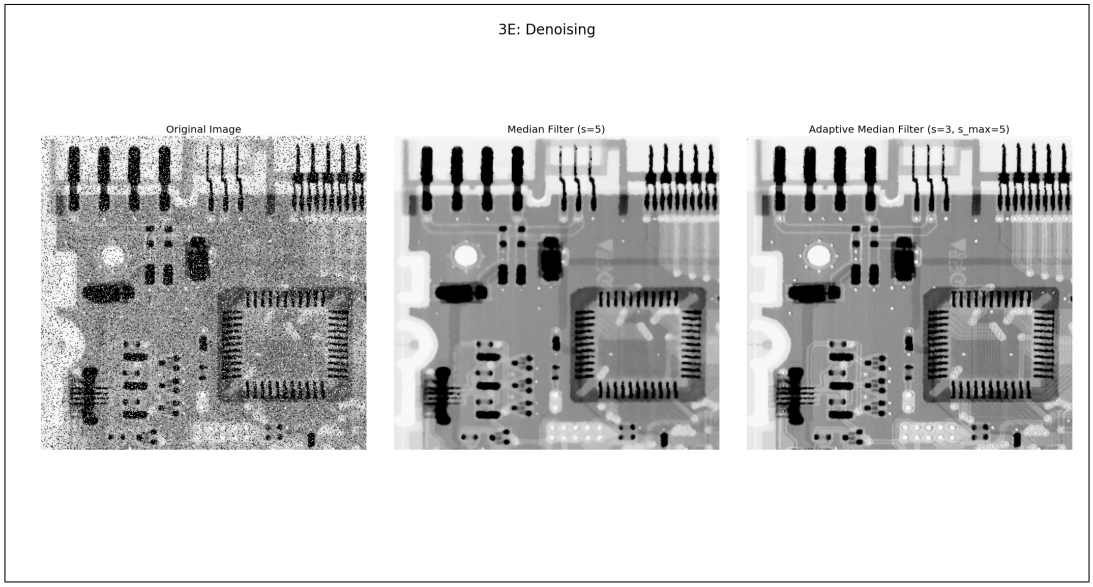

Figure 8: Result using an Adaptive Median Filter with starting filter region 3x3 and maximum filter region 5x5

Because the salt-and-pepper noise is not very dense ($P_a \approx P_b \approx 0.09$) the maximum filter region is set low at 5x5. In the result we see that all the noise is removed except for a couple of outliers on the boundary of the image. This might be because my implementation shrinks the filter region at the image boundaries and it could probably be fixed with a better solution. Some of the edges are a little bit distorted, but the details are better preserved than in the standard median filter (for example on the white circle in the upper left corner). The cost of this improvement is not very large when it comes to runtime of the algorithm, since most of the computation time is spent on calculating the median, which is done in both filters. The standard median filter finishes in 20 seconds, while the adaptive median filter has a 23 seconds runtime.

F)

To remove the periodic noise found in the last image I naturally decided to filter the image in the frequency domain, since this is where the noise is best detected and isolated for removal. The noise can be seen as a conjugate impulse pair on the power spectrum of the image, therefore I chose to use a notch reject filter to remove this impulse pair with minimal loss. A notch reject filter can reject predefined neighborhoods by taking the product of highpass filters whose center is shifted to the center of the notch. [11]

$$H_{NR}(u,v) = \prod_{k=1}^{Q} H_k(u,v) H_{-k}(u,v)$$

where $H_k(u,v)$ and $H_{-k}(u,v)$ are highpass filters whose centers are at (u_k, v_k) and $(-u_k, -v_k)$

respectively.

A single notch with a butterworth highpass filter pair is used in my case

$$H_{NR}(u,v) = [\frac{1}{1+[D_{0k}/D_k(u,v)]^{2n}}][\frac{1}{1+[D_{0k}/D_{-k}(u,v)]^{2n}}]$$

where $D_k(u,v)$ and $D_{-k}(u,v)$ are euclidean distances from point (u,v) to point (u_k, v_k) and $(-u_k, -v_k)$ respectively, n is the order and D_{0k} is the cutoff frequency of the notch.

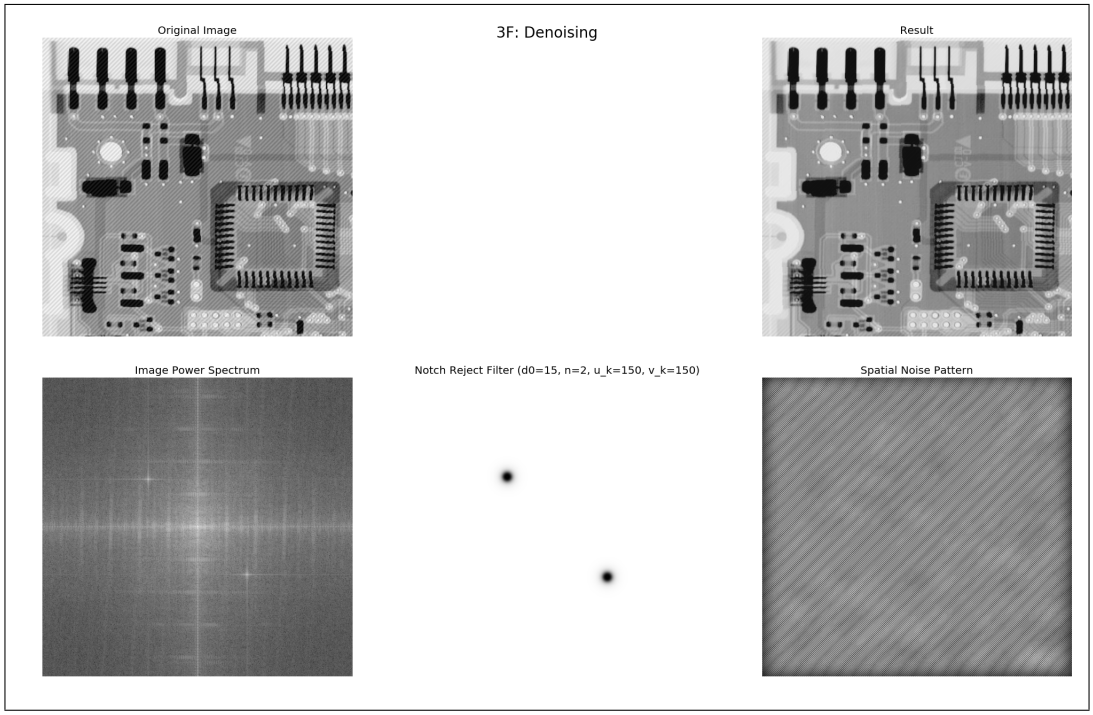

Figure 9: Result using Notch Reject Filter with a Butterworth Highpass Filter pair of order 2 with cutoff frequency 15 and centers at (150,150) and (-150,150)

By using a Notch Pass Filter on the same regions we rejected the noise, we can extract and examine the spatial noise pattern which is removed. We can see that using a Notch Reject Filter almost completely removes the periodic noise with minimal to no loss in the original image. There are some traces of the noise on the image boundary which seems to be a side effect of zero-padding the image, but the result is close to perfect overall.

References

[1] Gonzales, Rafael C. and Woods, Richard E. *Digital Image Processing*. Pearson Education, Third Edition, 2010. [1] p. 142, [2] p. 148, [3] p. 170, [4] p. 184, [5] p. 342-343, [6] p. 352-353, [7] p. 279-285, [8] p. 271-272, [9] p. 295-298, [10] p. 354-356, [11] p. 316-319,

All original and result images used is appended.

Source code for the problem and filter implementations is appended.