EQ2330 Image and Video Processing - Project 1

Stefano Imposcopi
imoscopi@kth.se

Denis Kulicek
kulicek@kth.se

November 24th, 2015

SUMMARY

The goal of this project is to investigate spatial and frequency domain image enhancement techniques. Problems under investigation are low-contrast images, and their enhancement with histogram equalization; applying spatial smoothing filters and order-statistics filters in order to remove unwanted noise from the image; and finally image deblurring by introducing Wiener filtering and image processing by multiplication in the frequency domain. Addressed problems are mainly related to the broad field of image enhancement.

Comparisons are made between different filters and visualizations like the histogram and the FFT magnitude spectrum and are used to qualitatively inspect properties of images and filters.
Also the problems of the implied periodicity of images in FFT computation on finite-size images is addressed and possible solutions are suggested.

INTRODUCTION

First out of three tasks is related to global histogram equalization to improve low contrast images. Then noise removal with low pass and median filters is investigated. Finally an implementation of the Wiener Filter for deblurring is presented.

SYSTEM DESCRIPTION AND RESULTS

We will be presenting examples on image typically used by Image processing community – Lena. The original picture together with its histogram can be seen on Figure 1.1

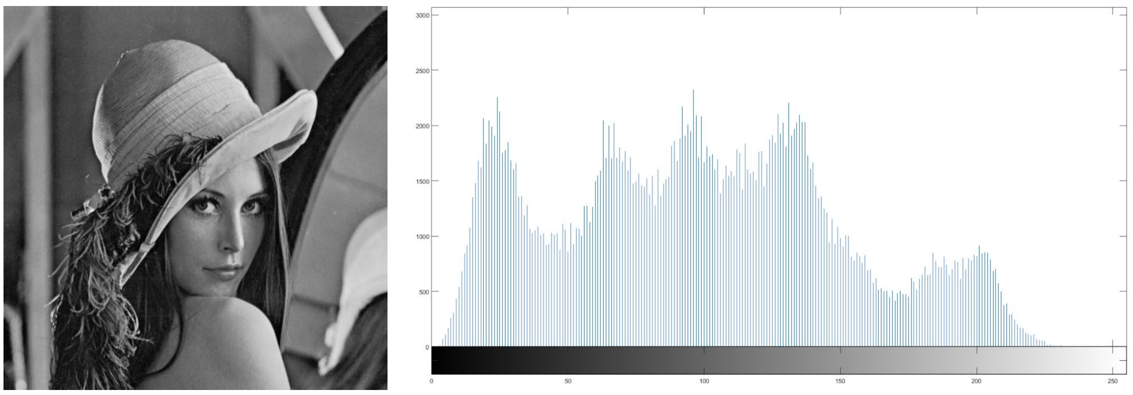

Figure 1.1 – Lena and histogram

As can be seen in the histogram, almost all the grayscale values are present, the image is taken under good illumination and the contrast is reasonable.
Next we simulate a low contrast image by linearly mapping the grayscale values to a subrange.

$$g(x,y) = round\,(\, a \cdot f(x,y) + b \,)$$

Setting `a=0.2` and `b=50` we map all the possible values to the small range `[50, 101]`. The result is a very low contrast image made of mostly gray values, as seen in Figure 1.2

To enhance the contrast we perform global histogram equalization, which maps each value k to its cumulative distribution, multiplied by the maximum value possible, 255 in our case

$$g_k = \frac{(L-1)}{MN} \sum_{j=0}^{k} n_j \qquad k \in [1, L-1]$$

Implementation of histogram equalization algorithm is presented in the appendix chapter.

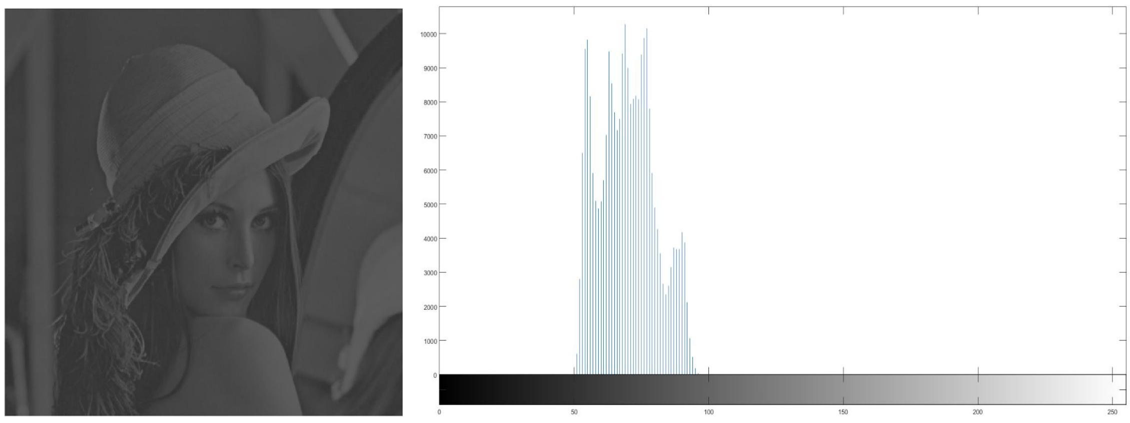

Figure 1.2 – Low contrast Lena and histogram

Why is histogram not flat after the equalization?

Although the perfect continuous histogram equalization should theoretically be a flat response over all intensities, this is not the case in practice. In the discrete case histogram equalization is a one to one mapping between grayscale values. Some close values can be merged together due to rounding, but overall the shape and modes of the histogram are preserved, just stretched over the whole dynamic range. In our case, this means filling the range of `[0, 255]` as seen on Figure 1.3.

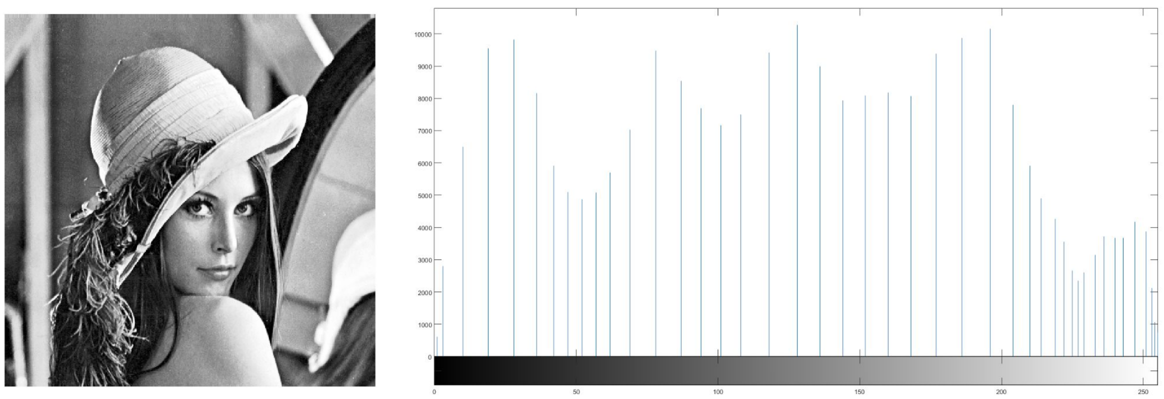

Figure 1.3. Equalized image representation, together with its histogram

What we can see here is that the reconstructed image seem to show much more information than the low-contrast image. Human visual cortex interprets contrast changes according to Weber's law, and therefore interprets small contrast changes as being less descriptive, hence carrying less information.

Next we investigate image-denoising spatial filters for different kinds of noise in the images. We used averaging 3x3 Low Pass Filter (LPF) as a spatial smoothing filter, and median 3x3 filter as an order-statistics filter.

Two types of noisy images $g_1(x,y)$ and $g_2(x,y)$ are investigated

1) $g_1(x,y) = f(x,y) + n(x,y)$ where $n(x,y)$ is **additive gaussian noise** with zero mean and variance 64
2) $g_2(x,y)$ is corrupted by **salt and pepper noise**, so some pixels are transformed into 0 or 255, the minimum and maximum grayscale value respectively. The probability of flipping a pixel is the same for both values and is equal to 5%.

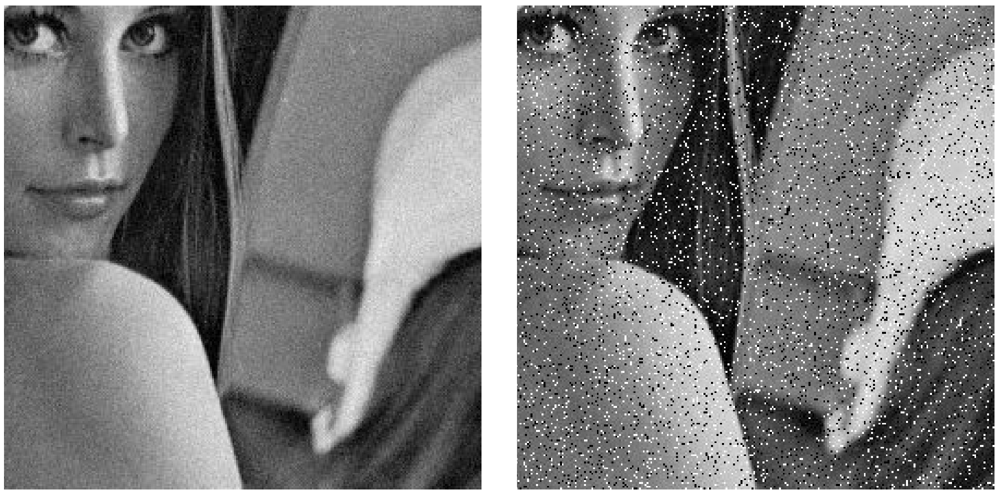

Figure 2.1 – bottom right corners of $g_1(x,y)$ and $g_2(x,y)$ respectively

The **LPF** is a linear filter, which can be implemented with the following convolution mask, in practice substituting every pixel using the average with its 8-pixel neighbourhood.

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

It is a separable filter which can be thus implemented using the one dimensional mask first across columns and then across rows.

```
(1/3)*[1 1 1]
```

The median filter is an order statistic filter that orders all the values in the 3x3 masks from smallest to largest and sets the median one as the value in the center of the mask.

Explain the difference between the mean filter and the median filter and their denoising effects on different noise types.

With Salt and Pepper noise in the image - as expected applying an averaging filter obtains a bad result, since the values 0 and 255 are at the extreme end of the dynamic range and they get blurred out over the other part of the image in the computation of the average. The median filter is a much better option since it can discriminate extreme values, hence black/white pixels (noise pixels) are very likely to be eliminated. Downside of this is that if most of the pixels where filter is ran are black/white, it will introduce image artifacts. This can be seen on Figure 2.2 (B), nearby middle of the picture.

For the additive gaussian noise both filters obtain acceptable results. The averaging filter works very well on smooth parts, but tends to blur the edges of the image. The median filter is more edge-preserving, which is a good property if we are interested in denoising without impacting the edges too much.

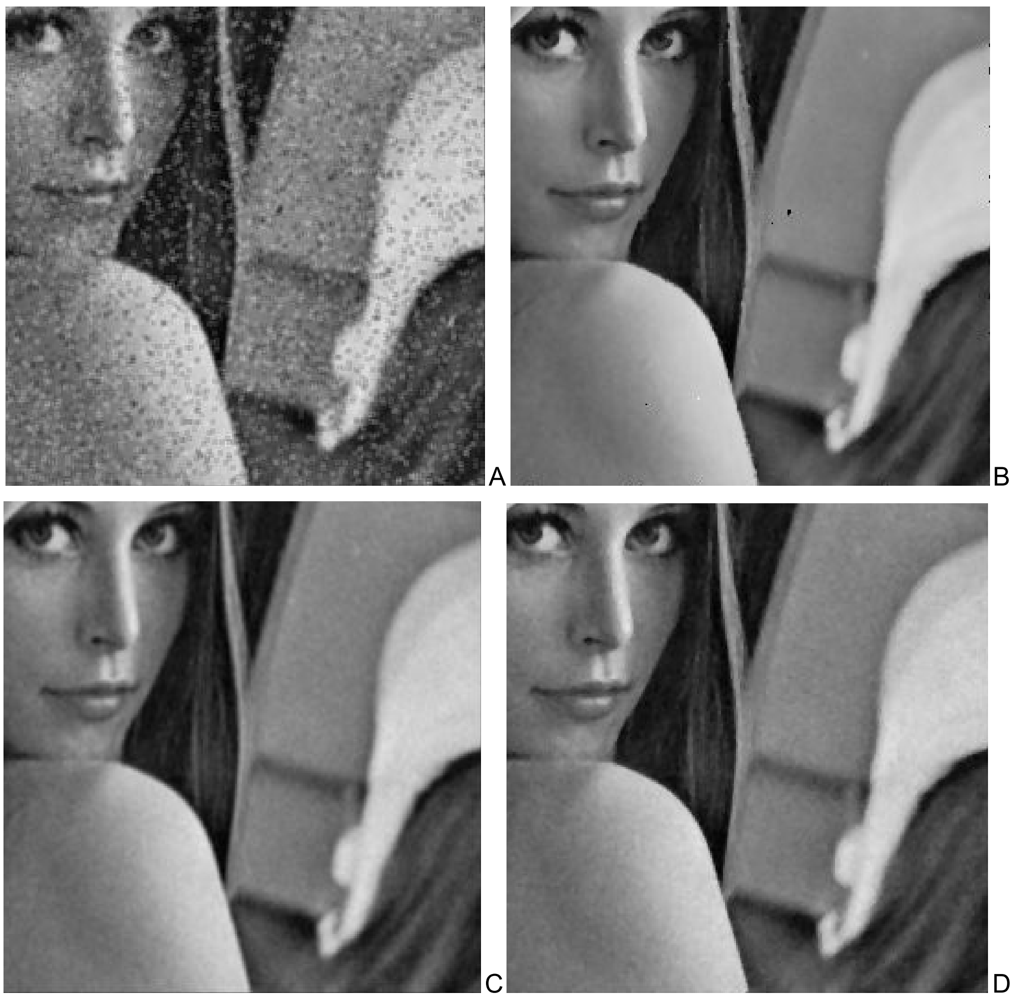

Figure 2.2 – Showing averaging filter vs median filter applied respectively on salt & pepper noise (A,B), and on gaussian noise (C,D)

The last experiment we performed is deblurring using filtering in the frequency domain. The blurred image $g(x,y)$ is obtained with the following equation

$$g(x,y) = h(x,y) * f(x,y) + n(x,y)$$

where $h(x,y)$ is the point spread function generating the blur, in our case a gaussian kernel blur, and $n(x,y)$ represents noise, always present in the image due to quantization and modeled by adding zero-mean additive gaussian noise with fixed variance to the clean image. Convolution becomes multiplication in the frequency domain, as can be visualized in Figure 3.1.

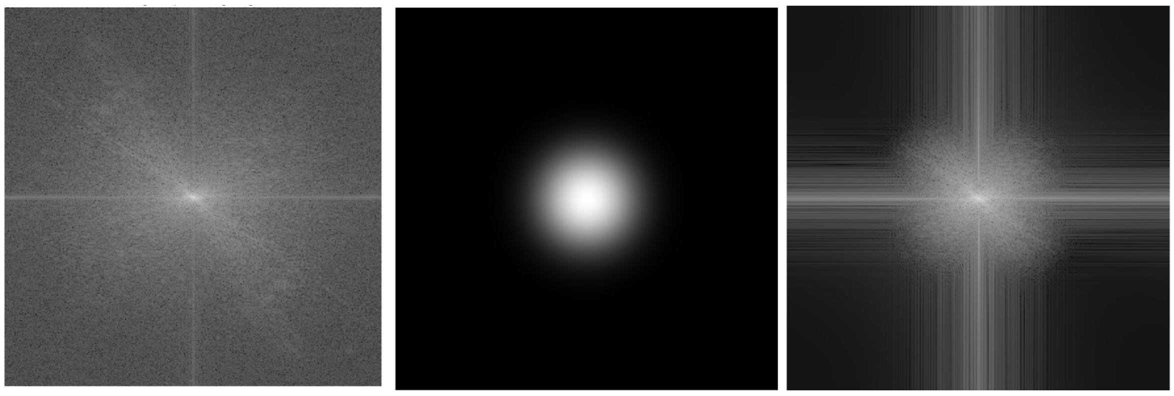

Figure 3.1 – From left to right - Multiplication of the original image magnitude spectrum with the gaussian blur filter H leading to a blurred image where high frequencies are lost.

The horizontal and vertical lines in the computed magnitude spectrums are non-idealities related to the finite size of the images, since ideally the DFT assumes infinite periodization of all the images and thus needs very high frequencies to represent the discontinuities at the boundaries. This can create ringing artifacts at the edges when using some algorithms. To reduce them the solution is applying a window function to smooth the edges and also apply smart padding techniques like the symmetric extension of the image, such that periodized images have smaller jumps in values at the edges.

To deblur the image we use the Wiener filter because it gives the best results in the presence of noise and generates less artifacts compared to inverse filtering (blind deconvolution).
Moreover the variance of the noise is known, so we can make an estimate of the Noise-To-Signal Ratio by assuming it constant.
The formula for deblurring in frequency domain and estimate $F(u,v)$ is the following

$$\left[\frac{H^*(u,v)}{|H(u,v)|^2 + S_\eta(u,v)/S_f(u,v)}\right]G(u,v)$$

where $H(u.v)$ is the DFT of the blur point-spread function and $S_n(u,v)/S_f(u,v)$ is the Noise-to-Clean-Signal Ratio. We simplify by assuming this quantity as a constant K and estimate it by a ratio of variances, using the knowledge about the blurred signal and the noise.

$$K = Var(\ n[x,y]\ )\ /\ Var(\ g[x,y]\ )$$

Assuming the NSR constant over all frequencies is a naive assumption, but still leads to much better results than inverse filtering, in the absence of knowledge about the power spectral density of the clean signal which is difficult to estimate..

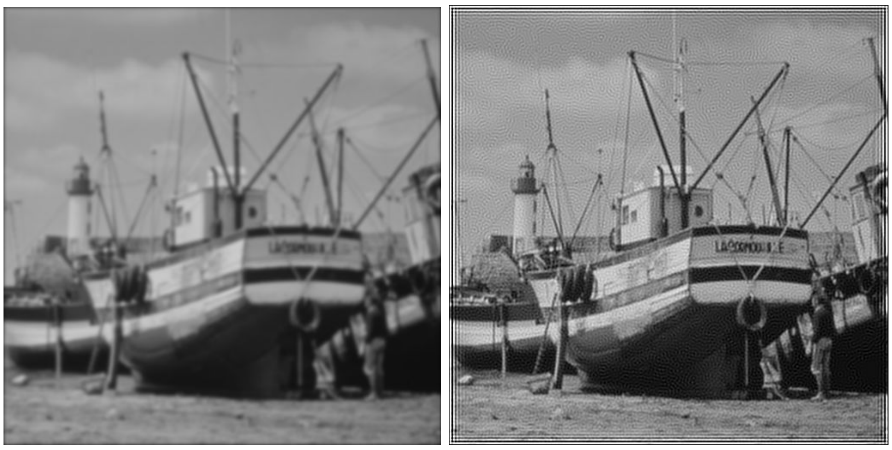

Figure 3.2 – Result of deblurring using Wiener Filter with fixed NSR K

To reduce ringing artifacts at the edges we periodize the image around the center with **symmetric extension** to obtain a size of 1024x1024 and an FFT of the same size. This already mitigated the problem of the implicit infinite periodicity of the images in the FFT computation.
The point spread function is padded with zeros to obtain an FFT of the same size.
Some ringing artifacts were anyway still present. To reduce them even further we apply a **windowing function** to taper the edges. Multiplication with a gaussian window with very large variance led to the best results. Also to reduce the residual noise we applied a 3x3 median filter as post processing. This gives a good tradeoff between blur removal and noise reduction.

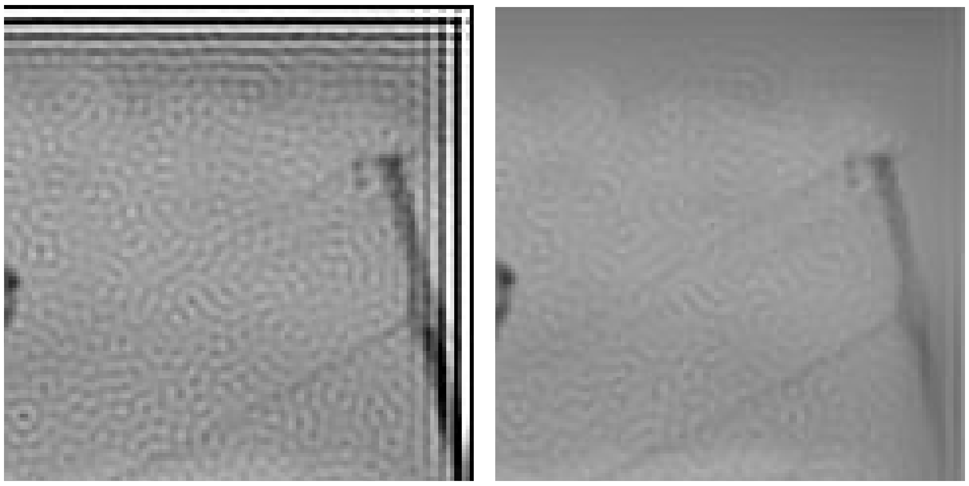

Figure 3.3 – No ringing and reduced residual noise in the right image after applying gaussian windowing and median filter post-processing.

CONCLUSIONS

From the histogram equalization task, we can conclude that even though some of the grayscale values were cut-off when linearly mapping them to only a subrange of values - The image information is still preserved. When histogram equalization is applied the perceptual quality of the image is greatly improved. It should anyway be noted that histogram equalization introduces values that were not originally present in the image and sometimes stretches values so much that the result is not natural. Also for some type of images global histogram equalization cannot work due to very different average values between different parts of the image. In that case a local histogram equalization would lead to much better results.

As expected the nonlinear median filter leads to much better results in the presence of salt and pepper noise characterized by extreme values. For additive gaussian noise averaging and median filter are comparable. Anyway the averaging LPF is very simple and leads to strong blurring of the edges of the images, which are important features for human perception. A combination of LPF and sharpening filters, for example adding the thresholded and weighted Laplacian or Sobel result, could lead to a much better linear denoising filtering technique where edges are better preserved.

Wiener filter for deconvolution of blurred images confirmed to be a good technique, much better than direct inverse filtering, because it can handle divisions by zero by taking into account the presence of noise in the blurred image. The biggest problem with the Wiener approach is the estimation of the Noise-to-Clean-Signal-Ratio and the need to assuming it constant over all frequencies in the absence of a priori-knowledge about the clean image. The parameters to be estimated are two power spectral densities to obtain an optimized result. Other techniques based on the optimization of a single parameter or iterative deblurring algorithms could be investigated to obtain a more automated deblurring algorithm that could work well with stronger blurring and other noise types, for example the regularized least-squares approach, based on the optimization of a single gamma parameter.

APPENDIX

The project was done collaboratively by both authors, while discussing the algorithms used throughout the process. Biggest learning outcomes came from experimenting with Wiener filter, and discussing the artifacts we were getting in reconstructed images - such as ringing and residual noise. Part of the code mentioned in the report is shown below, and the remaining source code is attached in a zip folder.

Histogram equalization

```
im = imread('lena512.bmp');
pdf_original = histcounts(im(:),[0:256]);
% reduce the contrast of the image by
% linearly mapping the range [0, 255] to [b, b+a*255]
a = 0.2;
b = 50;
% these values map [0,255] to [50,101]
for i=1:size(im,1)
    for j=1:size(im,2)
        im_lowc(i,j) = min(max(a*im(i,j) + b, 0), 255);
    end
end
```

Wiener filter for deblurring

```
function [f f_med] = deblur(g, b, n_var)
%Uses Wiener Filtering to deblur a noisy and blurred image
% g in the noisy and blurred image
% b is the convolution kernel for image deblurring
% var is the variance of the noise present in f
% f is the deblurred image
% f_med is an optional alternative with a 3x3 median filter post-processing
%   to remove residual noise after deblurring

im_var = var(g(:)); %used to estimate the NSR
K = n_var / im_var; %NSR ratio estimation assuming it is constant

% taper the edges of the image to reduce ringing
PSF = fspecial('gaussian',60,10);   %large gaussian window is used
g_taper = edgetaper(g,PSF);

% Pad with zeros to avoid artifacts and increase fft resolution
H = fft2(b, 1024, 1024); %fft of blur
H_mag = H.*conj(H);

filter = conj(H) ./ (H_mag + K);

% use symmetric padding to have the same fft resolution as H (1024 fft bins)
g_padded = padarray(g_taper,[256 256], 'symmetric');

out_freq = filter .* fft2(g_padded);

f = ifft2(out_freq);
% Apply 3-by-3 median filter to remove residual noise
f_med = medfilt2(f,[3 3]);
% crop only one replica of the image
f = f(249:760,249:760);
f_med = f_med(249:760,249:760);

end
```

References

[1] Rafael C. Gonzalez and Richard E. Woods, Digital Image Processing, Prentice Hall, 2nd ed., 2002