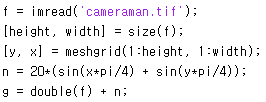
Homework 3: Image Restoration

# Periodic Noise Removal

## Add a periodic noise to a clean image



## Remove the periodic noise

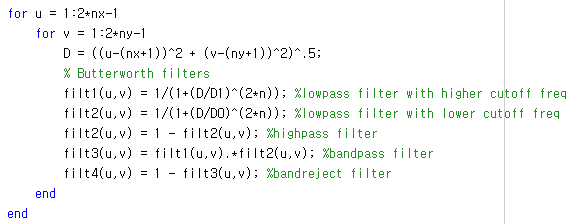
### Butterworth band reject filter

With this filter, frequencies at the center of the frequency band are completely blocked and frequencies at the edge of the band are attenuated by a fraction of the maximum value. The Butterworth filter does not have any sharp discontinuities between passed and filtered frequencies.

The centered FFT is filtered by band reject function:



Where D0 is the center of the frequency , W is the width of the frequency band, D is the distance between a point (u,v) in the frequency domain and the center of the frequency rectangle, and n is the order of the Butterworth filter.



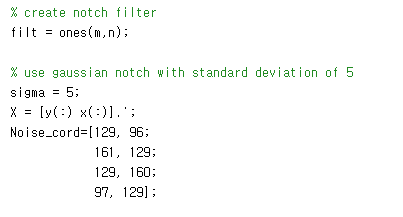
### Notch reject filter

A notch reject filter rejects frequencies in a predefined neighbourhood about the center of the frequency rectangle. Notch reject filters are constructed as products of highpass filters whose centers have been translated to the centers of the notches. The general form is

Where Hk(u,v) and H-k(u,v) are highpass filters whose centers are at (uk,vk) and (-uk,-vk).

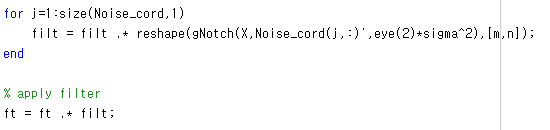
Implementation in matlab using Gaussian notch filter with standard deviation (sigma) is 5





For different image with different noise, we can extract noise coordinates from image spectrum in frequency domain.

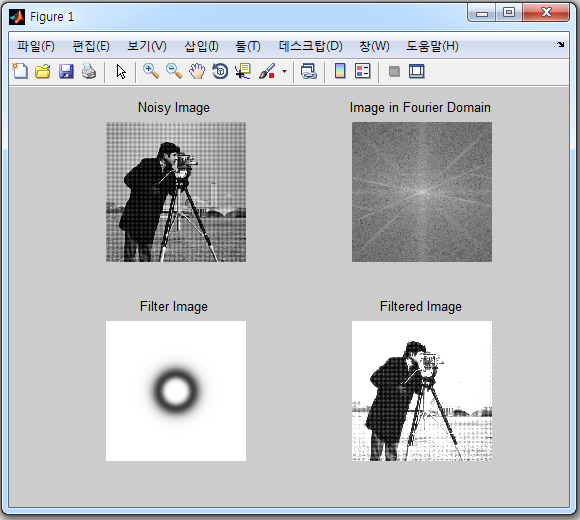
And filter the image with notch filter



## Results

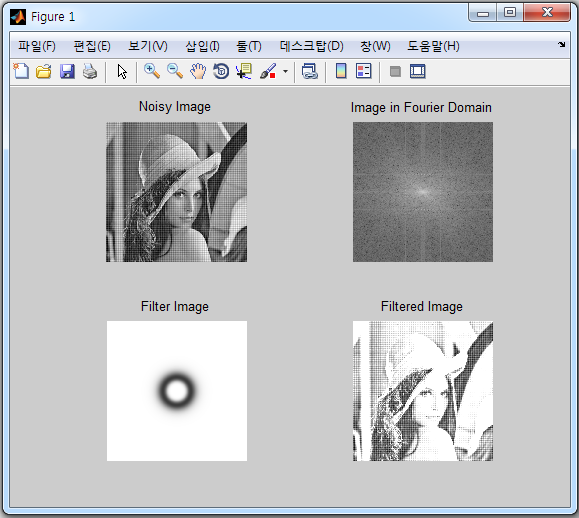
### Butterworth band reject filter

Cameraman image:



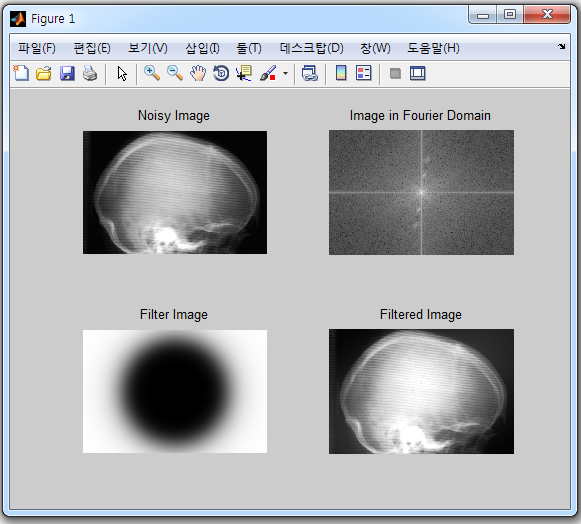


Lena image:





Im1 image:



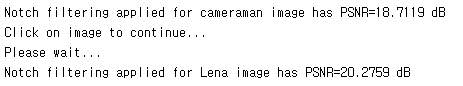
Im2 image:



### Gaussian notch reject filter

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |

Calculating PSNR when applying Notch filter



From the results, we can conclude that Notch filter is better than Butterworth band reject filter.

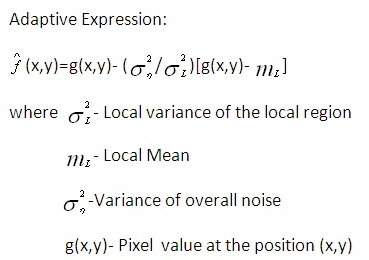
# Simple Image Denoising

## Add AWGN

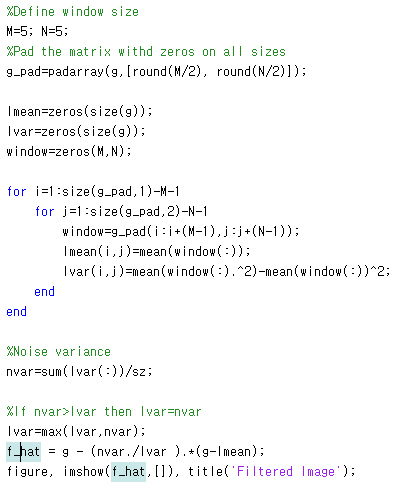
Because awgn() function is in Communications System Toolbox, I don’t have this toolbox so I cannot use this function directly. Instead, add white Gaussian noise via rand() function with standard deviation (sigma) is 10 as below:



## Applying adaptive, local noise reduction filter



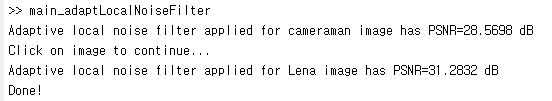
Implementing this filter in matlab as below:



Adaptive, local noise reduction filter results

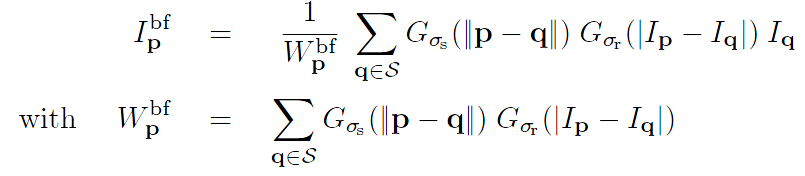
|  |  |
| --- | --- |
|  |  |
|  |  |

According to the experiment, window size (M=N=5) results in highest quality

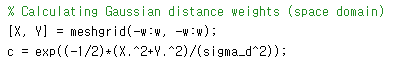


## Applying Bilateral Filter

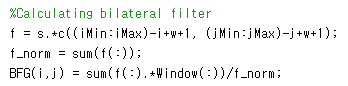
Bilateral Filter is expressed as below equation



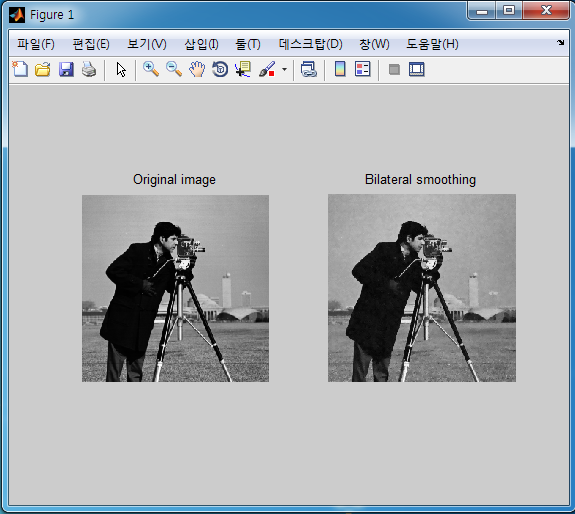
Combined domain and range filtering will be denoted as bilateral filtering. It replaces the pixel value at p with an average of similar and nearby pixel values. In smooth regions, pixel values in a small neighbourhood are similar to each other, and the bilateral filter acts essentially as a standard domain filter, averaging away the small, weakly correlated differences between pixel values caused by noise.

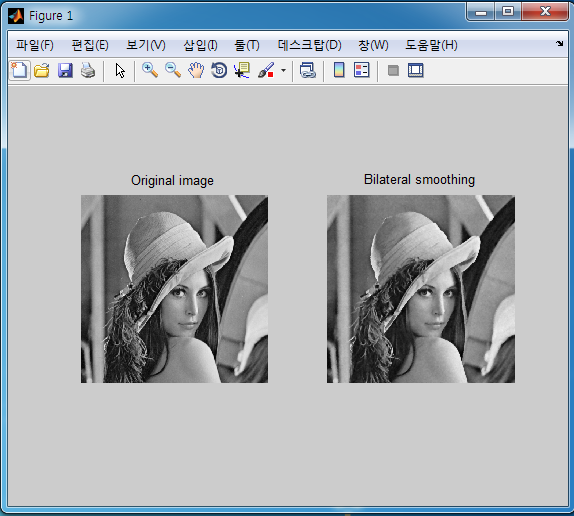




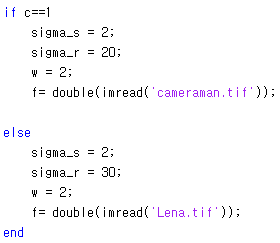


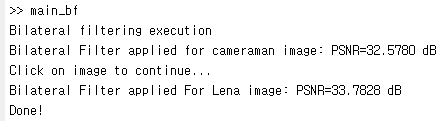
Bilateral Filter results





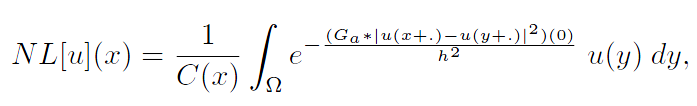
In order to achieve the best PSNR, adjust the window size and two parameters (sigma\_s and sigma\_r) use in BF as below





## Applying the non-local mean filter (NLM)

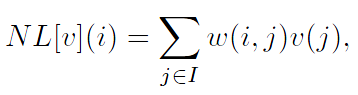
The NL-means algorithm, which is defined by the formula



Where 

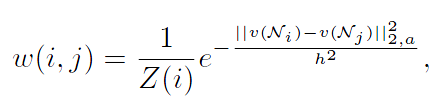
Is a normalizing constant, Ga is a Gaussian kernel and h acts as a filtering parameter. This formula amounts to say that the denoised value at x is a mean of the values of all points whose Gaussian neighbourhood looks like the neighbourhood of x.

Given a discrete noisy image, the estimated value NL[v](i), for a pixel I, is computed as a weighted average of all the pixel in the image,

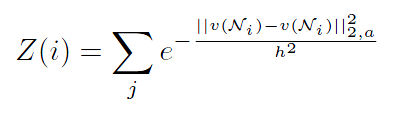


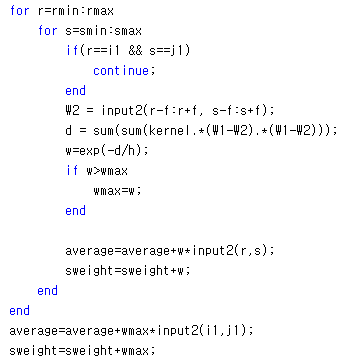
Where the family of weights {w(I,j)}j depend on the similarity between the pixels I and j

The weights are defined as



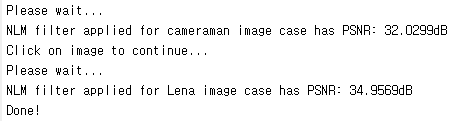
Where Z(i) is the normalizing constant





NLM filter results

|  |  |
| --- | --- |
|  |  |
|  |  |



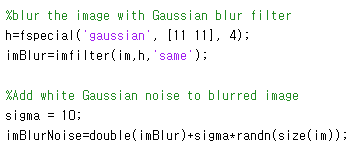
# Simple Image Restoration

## Blurring and adding noise to image

Gaussian blur kernel: 11x11 window,

G(x,y) = (1/(sqrt(2\*pi)\*σ))\*exp(-((x-5)2 + (y-5)2)/(2σ2)), with σ =4.0.

And then add AWGN to blurred image



## Restore the corrupted image using Wiener filter

Wiener filter is expressed by the equation

G(u,v)

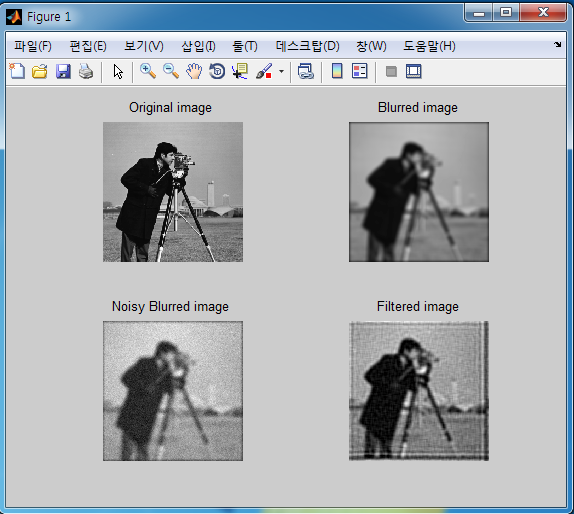
Where K is a specified constant that is added to all terms of |H(u,v)|2  and

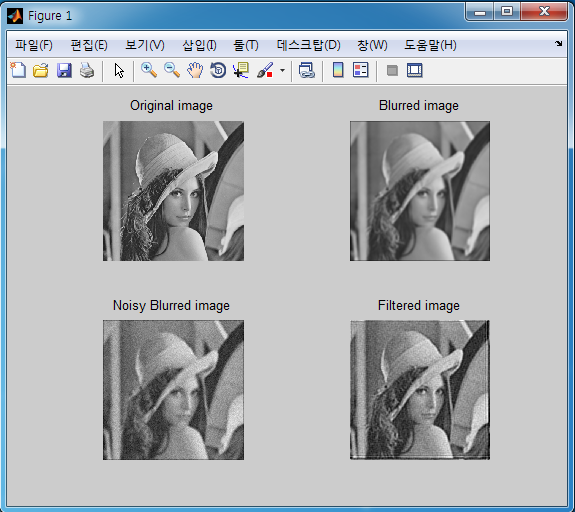
K=1/SNR.

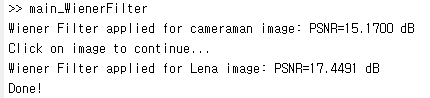
This filter is implemented as follow



Wiener filter results







## Constrained Least Squares filter

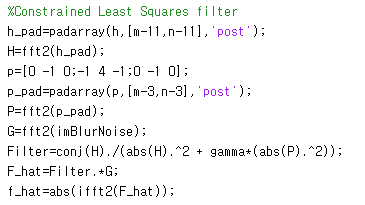
In frequency domain, the equation of this filter is



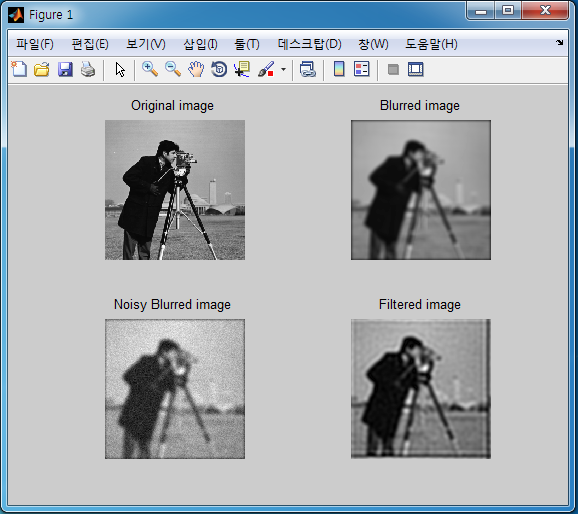
Where ϒ is the parameter to be adjusted (ϒ=0 -> inverse filtering), and P(u,v) is the fourier transform of the function

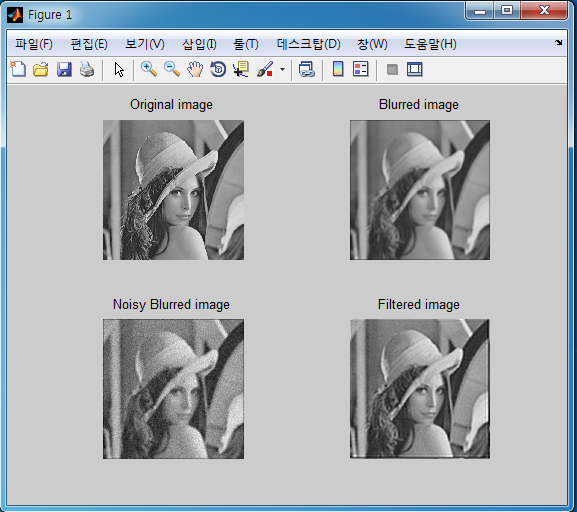


This filter is implemented as follow:



Constrained Least Squared filter results







Subjectively, we can see the same results from two filters.

Objectively, CLS gives us a better result than Wiener filter.