# 2. Wiener filter

December 5, 2020

## 1 Theory behind Weiner filter

Let's remind ourselves of the model of distortions in the freqency domain:

$$\boldsymbol{f} = \boldsymbol{H_d s} + \boldsymbol{n}$$

Now let's write a formula for the power of the noise.

Energy spectrum of the signal is computed by taking the absolute value of the absolute value of the result of a fourier transformation of the signal and squaring it: $|\hat{x}|^2$. Where $\hat{x}$ is a result of the fourier transformation.

Based on the model, the estimate of the noise power will be:

$$|\boldsymbol{H_d s} - \boldsymbol{f}|^2 = |-\boldsymbol{n}|^2 = |\boldsymbol{n}|^2$$

Now let's suppose that our filter $\boldsymbol{H}$ produces a signal estimate $g = \boldsymbol{H f}$.

If the filter is perfect (our goal always) then $g$ is equal to $s$. That means that our perfect result of the perfect fiter would satisfy this equation as well:

$$|\boldsymbol{H_d g} - \boldsymbol{f}|^2 = |\boldsymbol{n}|^2$$

Let's try to find the perfect signal $g$ by solving the above equation. Unfortunately we do not know the noise $n$. So let's set it to 0. Why not?

In that case finding a best fitting $g$ would mean minimizing the following term:

$$|\boldsymbol{H_d g} - \boldsymbol{f}|^2$$

Taking the derivative with respect to the estimate vector, we obtain:

$$\frac{\partial(\boldsymbol{H_d g} - \boldsymbol{f})^T(\boldsymbol{H_d g} - \boldsymbol{f})}{\partial g} = 2\boldsymbol{H_d}^T(\boldsymbol{H_d g} - \boldsymbol{f})$$

Solving for derivative = 0 we obtain the following solution:

$$\boldsymbol{g} = \boldsymbol{H_d}^{-1}\boldsymbol{f}$$

This is the familiar inverse filter, the deterministic solution, appropriate for the zero noise case. But as it was shown in the chapter 1, this results in significant noise implification.

No luck!

But what we can do is to use our knowledge that a best fitting $g$ is going to satisfy:

$$|\boldsymbol{H_d g} - \boldsymbol{f}|^2 = |\boldsymbol{n}|^2$$

We don't know what $n$ is so we cannot solve it turns out that we can use it as a constrain for optimization of another term and obtain result this way.

The term to minimize that we will use is as follows:

$$J(\boldsymbol{g}) = |\boldsymbol{Q g}|^2$$

The criterion matrix $\boldsymbol{Q}$ is an LSI system chosen to select for the undesirable components of the estimate $g$.

So our problem is now to minimize the $J(\boldsymbol{g})$ while keeping the constrain $|\boldsymbol{H_d g} - \boldsymbol{f}|^2 = |\boldsymbol{n}|^2$.

This can be done using a method of Lagrange multipliers. How this is done is beyond a scope of this notebook.

The result is something like this:

$$\boldsymbol{g} = [\boldsymbol{H}_d^T \boldsymbol{H} + \gamma \boldsymbol{Q}^T \boldsymbol{Q}]^{-1} \boldsymbol{H}_d^T \boldsymbol{f}$$

Where $\gamma$ is langrange multiplier that we have to find. Unfortunately there is no closed form for this so in practice we set a value of this constant empirically.

After some more math magic we obtain a formula for the frequency of the optimal filter:

$$H(f) = \frac{H_d(f)^*}{|H_d(f)|^2 + \gamma |Q(f)|^2}$$

They are few choices for $Q(f)$ and $\gamma$.

First choice is $Q$ that does not modify anything and use a contant $\gamma$ to control the noise.

$$H(f) = \frac{H_d(f)^*}{|H_d(f)|^2 + \gamma}$$

The second choice is to set $\gamma |Q(f)|^2$ to $\frac{P_n(f)}{P_s(f)}$ which corresponds to a SNR of the signal.

$$H(f) = \frac{H_d(f)^*}{|H_d(f)|^2 + \frac{P_n(f)}{P_s(f)}}$$

They are also choices that involve difference operators.

Here we will use the first version as I'm lazy (:

Also it works quite well with gaussian noise.

## 2 Example of the Winer filter

First let's load image and distort it.

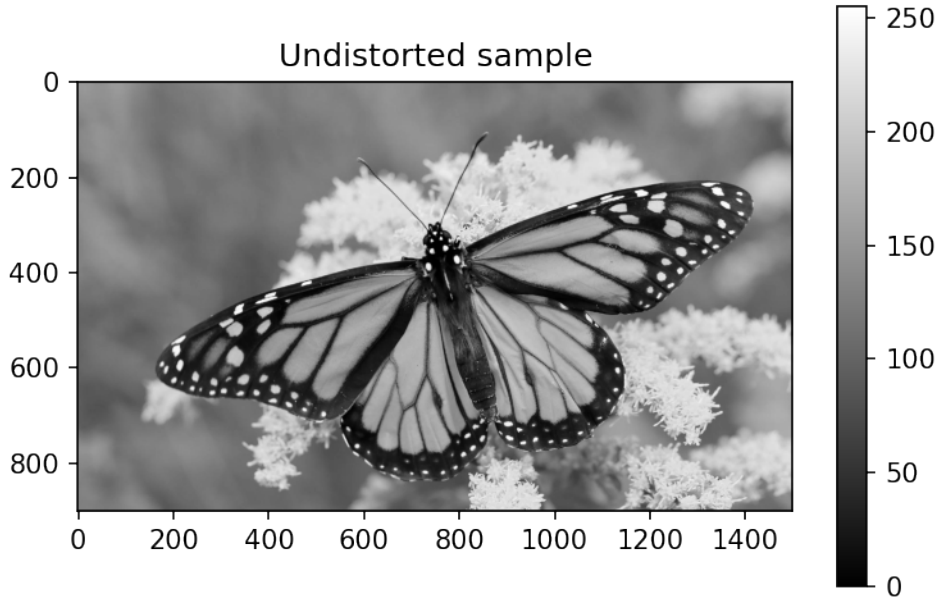
[1]:
```
%matplotlib inline
import cv2
import matplotlib.pyplot as plt
import matplotlib
matplotlib.rcParams["figure.dpi"] = 150
```

[2]:
```
from scipy.signal import gaussian
import numpy as np

def gaussian_kernel(kernel_size = 3):
        h = gaussian(kernel_size, kernel_size / 3).reshape(kernel_size, 1)
        h = np.dot(h, h.transpose())
        h /= np.sum(h)
        return h
```

[3]:
```
sample = cv2.imread("../samples/monarch.jpg", cv2.IMREAD_GRAYSCALE)
plt.title("Undistorted sample")
plt.imshow(sample, cmap='gray')
plt.colorbar()
```

[3]:
```
<matplotlib.colorbar.Colorbar at 0x7f767ae5bac8>
```



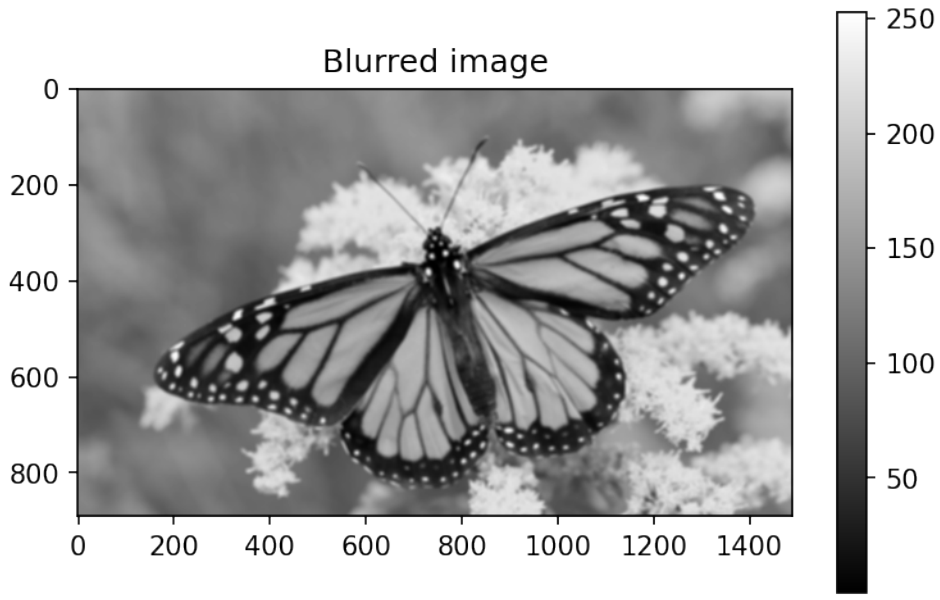
[4]:
```
from scipy.signal import convolve2d
from scipy.signal import gaussian

def gaussian_kernel(kernel_size = 3):
        h = gaussian(kernel_size, kernel_size / 3).reshape(kernel_size, 1)
        h = np.dot(h, h.transpose())
        h /= np.sum(h)
        return h

def blur(img, kernel_size = 3):
        dummy = np.copy(img)
        h = gaussian_kernel(kernel_size)
        dummy = convolve2d(dummy, h, mode = 'valid')
        return dummy
```

[5]:
```
blurred_img = blur(sample, kernel_size = 10)

plt.title("Blurred image")
plt.imshow(blurred_img, cmap='gray')
plt.colorbar()
#plt.savefig('2_a_bit_blurred.png', dpi=300)
```



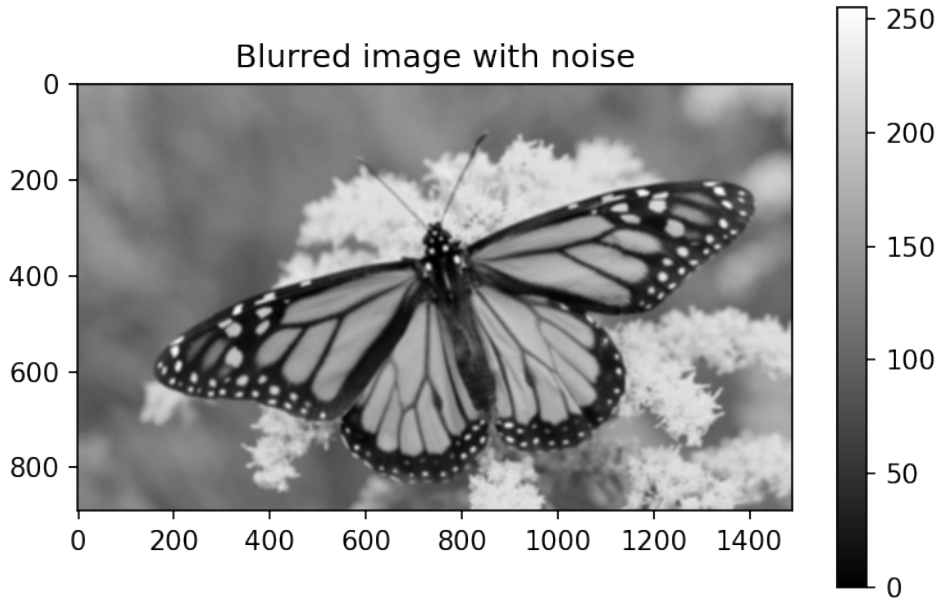

[6]:
```
def add_gaussian_noise(img, sigma):
        gauss = np.random.normal(0, sigma, np.shape(img))
```

```
        noisy_img = img + gauss
        noisy_img[noisy_img < 0] = 0
        noisy_img[noisy_img > 255] = 255
        return noisy_img
```

[7]:
```
noisy_img = add_gaussian_noise(blurred_img, sigma = 3)

plt.title("Blurred image with noise")
plt.imshow(noisy_img, cmap='gray')
plt.colorbar()
```

[7]:
```
<matplotlib.colorbar.Colorbar at 0x7f767ab59e80>
```




Now let's estimate blurr kernel.

[8]:
```
kernel = gaussian_kernel(10)
```

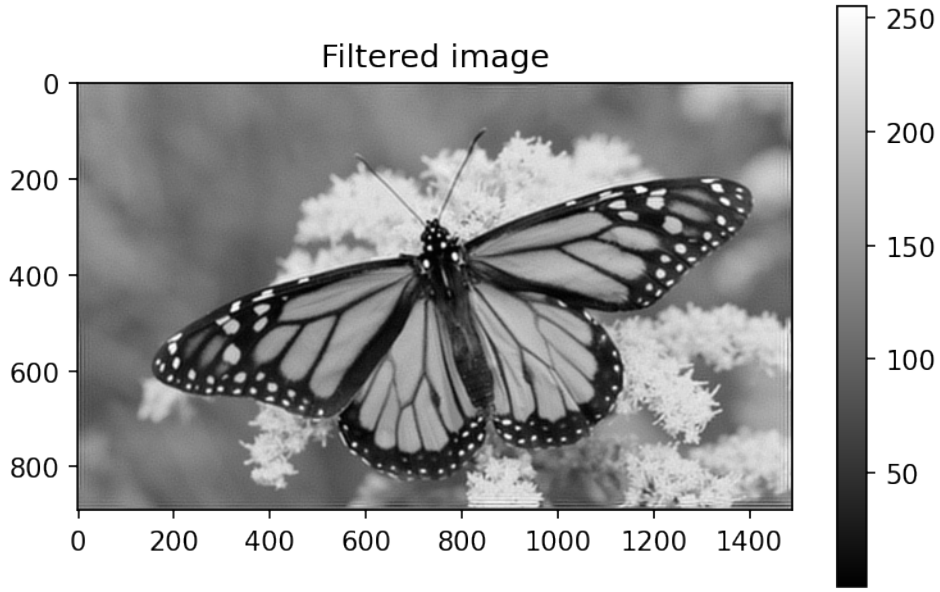
And now let's implement the weiner filter as presented on the above formula.

[9]:
```
from numpy.fft import fft2, ifft2

def wiener_filter(img, kernel, K):
        kernel /= np.sum(kernel)
        dummy = np.copy(img)
        dummy = fft2(dummy)
        kernel = fft2(kernel, s = dummy.shape)
        kernel = np.conj(kernel) / (np.abs(kernel) ** 2 + K)
```

```
        dummy = dummy * kernel
        dummy = np.abs(ifft2(dummy))
        return dummy
```

[10]:
```
filtered_img = wiener_filter(noisy_img, kernel, K = 0.01)
filtered_img[filtered_img>255] = 255
plt.title("Filtered image")
plt.imshow(filtered_img, cmap='gray')
plt.colorbar()
#plt.savefig('2_filtered1.png', dpi=300)
```

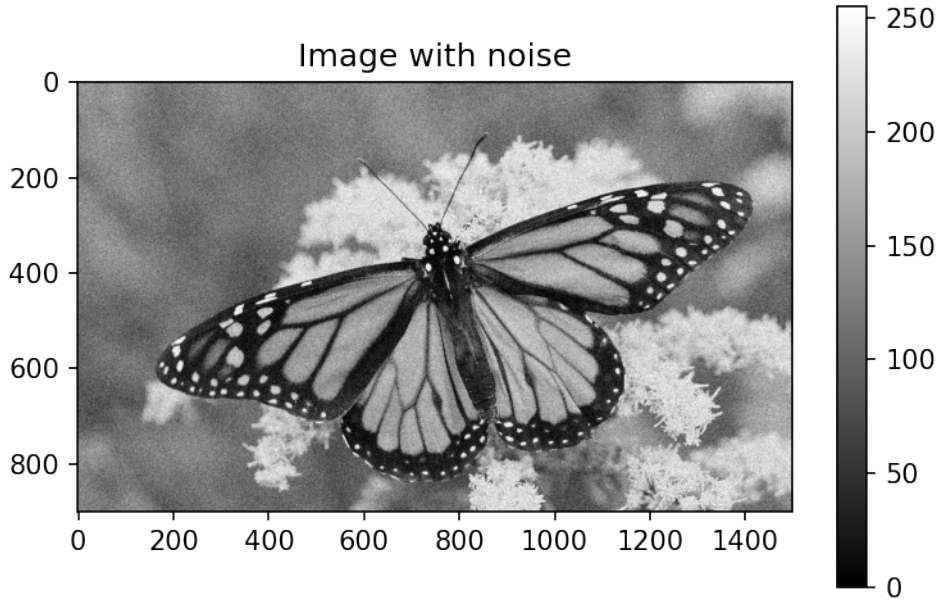



Result can be improved by using SNR but it is difficult to estimate.

Weiner filter is also good at removing the noise from the image.

[11]:
```
noisy_img = add_gaussian_noise(sample, sigma = 30)

plt.title("Image with noise")
plt.imshow(noisy_img, cmap='gray')
plt.colorbar()
#plt.savefig('2_alot_noise.png', dpi=300)
```




[12]:
```
kernel = gaussian_kernel(8)
filtered_img = wiener_filter(noisy_img, kernel, K = 3)
filtered_img[filtered_img>255] = 255
plt.title("Filtered image")
plt.imshow(filtered_img, cmap='gray')
plt.colorbar()
#plt.savefig('2_filtered2.png', dpi=300)
```