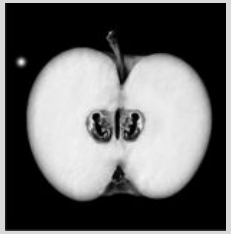
Lab 07

Basic Image Processing
Fall 2018

Image degradation and restoration

A typical imaging system

Original image



The image is somewhat modified during capturing

Some noise might also appear

Captured image

DEGRADATION block diagram

Image degradation and restoration

DEGRADATION block diagram

"Degradation happens"

We don't want a reduced quality image but that's the best the system can capture.

Quality loss happens due to motion, out-of-focus lenses, transmission and quantization errors etc.

Usually it is not possible (or affordable) to eliminate the causes of the degradation. Instead, we use restoration approaches to get back the original image from the low quality one.

Let’s take it to the next level!

DEGRADATION block diagram

Spatial domain

$$x(n_1, n_2) \otimes h(n_1, n_2) + n(n_1, n_2) = y(n_1, n_2)$$

The system is LSI (*linear space-invariant*) so we can Fourier transform it to the frequency domain:

Frequency domain

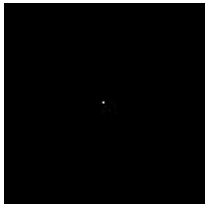
$$X(\omega_1, \omega_2) \cdot H(\omega_1, \omega_2) + N(\omega_1, \omega_2) = Y(\omega_1, \omega_2)$$

Point spread function & Optical transfer function

We can describe H with an image-pair. Let's create a well-known image and send it through the imaging system. The response will be a modified image.
If we assume no noise then the input-output pair perfectly describes the H.

an arbitrary input   the response of the system

If the input image contains only one point (a pixel) then we call the response the *point spread function* and we can use this single image to describe the system.

a "point"   the response is the *point spread function*

Point spread function & Optical transfer function

The point spread function can be used to realize the system. An image convolved with the point spread function is the response of the *H* block.

Spatial domain

an arbitrary input | convolved with | the PSF | is | the response of the system

$\otimes$

$=$

$x(n_1, n_2) \quad \otimes \quad h(n_1, n_2) \quad = \quad y(n_1, n_2)$

Frequency domain

$$X(\omega_1, \omega_2) \quad \bullet \quad H(\omega_1, \omega_2) \quad = \quad Y(\omega_1, \omega_2)$$

We call the Fourier transform of the PSF "**optical transfer function**" (OTF)

Image degradation and restoration

DEGRADATION block diagram

Since we have the model of the degradation system, we can build its inverse.

RESTORATION block diagram

Now we just have to find n and H^{-1} and we are ready :)

Image degradation and restoration

RESTORATION block diagram

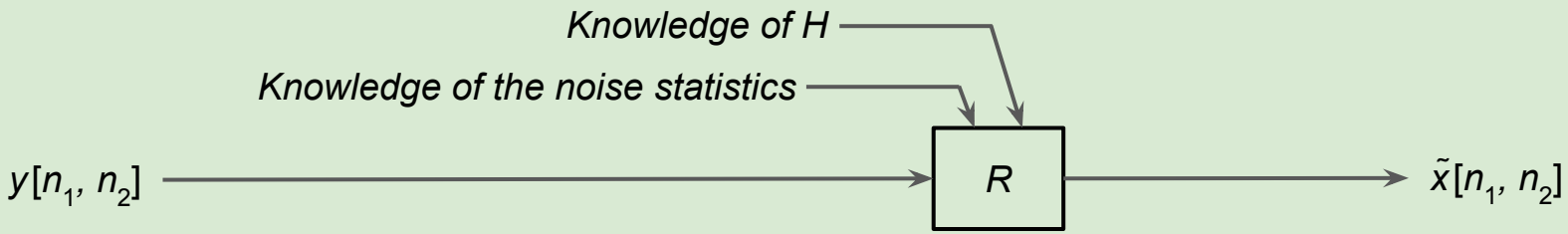
In practice it is impossible to find the exact n ⇒ subtraction cannot be done.
However, we can do some ***noise statistics*** or ***assume*** the type of noise.

In practice we don't know the exact H either.
However, we can do ***measurements*** or we can ***estimate*** the system. Also, we have to ***deal with the noise*** here.

Therefore, instead of using H^{-1} it is very common to create an R 'recovery' system.

Image degradation and restoration

RESTORATION block diagram



Depending on the restoration approach the ***R*** system can be…

Inverse Filter	$\underset{x}{\operatorname{argmin}} \left(\|y - Hx\|^2 \right)$	$R(\omega_1, \omega_2) = \dfrac{1}{H(\omega_1, \omega_2)}$
Constrained Least Square	$\underset{x}{\operatorname{argmin}} \left(\|y - Hx\|^2 + \alpha \|Cx\|^2 \right)$	$R(\omega_1, \omega_2) = \dfrac{H^*(\omega_1, \omega_2)}{\|H(\omega_1, \omega_2)\|^2 + \alpha \|C(\omega_1, \omega_2)\|^2}$
Wiener Filter	$\underset{x}{\operatorname{argmin}} \left(\mathbb{E} \left[\|x - \tilde{x}\| \right] \right)$	$R(\omega_1, \omega_2) = \dfrac{H^*(\omega_1, \omega_2)}{\|H(\omega_1, \omega_2)\|^2 + \dfrac{P_{NN}(\omega_1, \omega_2)}{P_{XX}(\omega_1, \omega_2)}}$

Now please

download the ‘Lab 07’ code package

from the

submission system

Exercise 1

Implement the function `degradation` in which:

- The function has three input and three output arguments.
- Inputs:
  - `x` the input image (grayscale, double, within range [0,1])
  - `kernel` the chosen kernel type as a string
  - `noise` the chosen noise type as a string
- Outputs:
  - `y` the degraded output image
  - `h` the kernel of the imaging system
  - `n` the values of the additive noise layer

Exercise description continues on the next slide...

Exercise 1 – continued

Valid kernel strings are `'average'`, `'gaussian'` and `'motion'`.

- If `'average'` kernel is chosen `h` should be a 5×5 averaging kernel.
- If `'gaussian'` kernel is chosen `h` should be a 9×9 Gaussian kernel with $\sigma = 1$
- If `'motion'` kernel is chosen `h` should be a motion kernel, `len=11`, `theta=30`

Use `fspecial` to produce the kernel matrices.

Exercise description continues on the next slide...

Exercise 1 – continued

Valid noise strings are `'off'`, `'white'` and `'pink'`.

Every `n` noise layer has to be the same size as the input image `x`.

- If `'off'` noise is chosen `n` should be filled with zeros.
- If `'white'` noise is chosen `n` should be filled with normally distributed random numbers (use `randn`). The variance should be $\sigma^2 = 0.001$
  (Multiply the result of `randn` by `sqrt(0.001)` to get the desired variance.)
- If `'pink'` noise is chosen `n` should be filled with numbers generated using the `pinknoise` function. Multiply the result by 0.05 to get a decent noise level.

Exercise description continues on the next slide...

Exercise 1 – continued

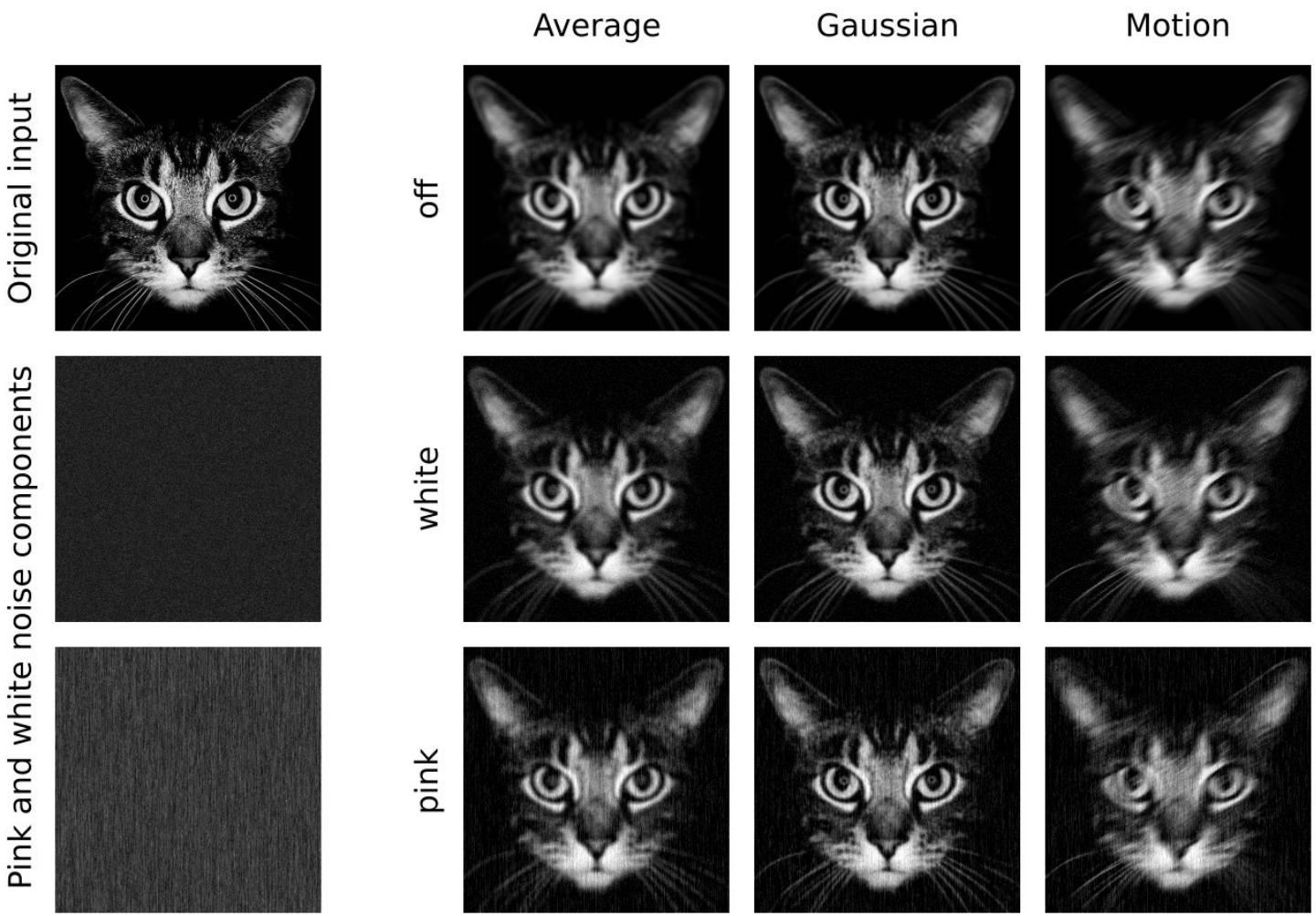
Use `if` statements and `strcmp` to check what type of kernel and noise is required.

Raise an error (using `error()`) if the kernel or noise name is invalid.

After creating the appropriate `h` and `n` matrices, apply the kernel to the input using `imfilter` with the options `'replicate'`, `'same'`, `'conv'`, and after this, add the noise layer to the filtered image.

$$y = x \otimes h + n$$

Execute `script1.m` and check your results!


Average
Gaussian
Motion
Original input
off
Pink and white noise components
white
pink

Exercise 2

Implement the function `inverse_filter` in which:

- The function has one input (`H`) which is the optical transfer function of the imaging system.
- The function have to return the OTF of the reconstruction system (`R`).

Implement the following formula:

$$R(\omega_1, \omega_2) = \frac{H^*(\omega_1, \omega_2)}{\left|H(\omega_1, \omega_2)\right|^2}$$

in which H^* means complex conjugate (use the `conj()` function in MATLAB)

Exercise 3

Implement the function `CLS_filter` in which:

- The function has three inputs:
  - `H` is the optical transfer function of the imaging system,
  - `alpha` is a regularization parameter, and
  - `C` is the optical transfer function of a high pass filter
- The function have to return the OTF of the reconstruction system (`R`).

Implement the following formula:

$$R(\omega_1, \omega_2) = \frac{H^*(\omega_1, \omega_2)}{|H(\omega_1, \omega_2)|^2 + \alpha |C(\omega_1, \omega_2)|^2}$$

in which H^* means complex conjugate (use the `conj()` function in MATLAB)

Exercise 4

Implement the function `wiener_filter` in which:

- The function has three inputs:
  - `H` is the optical transfer function of the imaging system,
  - `P_NN` is the power spectrum of the noise (or variance if white noise)
  - `P_XX` is the power spectrum of the signal (or variance if white noise)
- The function have to return the OTF of the reconstruction system (`R`).

Implement the following formula:

$$R(\omega_1, \omega_2) = \frac{H^*(\omega_1, \omega_2)}{|H(\omega_1, \omega_2)|^2 + \frac{P_{NN}(\omega_1, \omega_2)}{P_{XX}(\omega_1, \omega_2)}}$$

in which H^* means complex conjugate (use the `conj()` function in MATLAB)

Some extra words about the Wiener filter

Wiener filter uses a stochastic approach. It smooths the additive noise and inverts the filtering simultaneously.

$$R(\omega_1, \omega_2) = \frac{H^*(\omega_1, \omega_2)}{|H(\omega_1, \omega_2)|^2 + \frac{P_{NN}(\omega_1, \omega_2)}{P_{XX}(\omega_1, \omega_2)}}$$

There is an inverse filtering and a noise smoothing part in the formula. For the latter purpose the filter takes into account the *power spectra* of the noise and the (original) signal.

Since in practice we don't have the X, we estimate P_{XX} based on Y.

Autocorrelation function & Power spectrum

The **Autocorrelation function (ACF)** of a 1D symmetric signal is the signal convolved with its complex conjugate:

$$R_{ff} = f \otimes f^*$$

In case of 2D images the *wide sense stationary property* holds and then thanks to the *Wiener-Khinchin theorem* the ACF can be calculated using the following steps:

Input image: $x(n_1, n_2)$

Fourier transform: $\{X(\omega_1, \omega_2)\} = \mathrm{DFT}\{x(n_1, n_2)\}$

Power spectrum: $P_{xx}(\omega_1, \omega_2) = X(\omega_1, \omega_2) \cdot X^*(\omega_1, \omega_2)$

Autocorrelation: inverse Fourier transform of the power spectrum:
$\{R_{xx}(d_1, d_2)\} = \mathrm{IDFT}\{P_{xx}(\omega_1, \omega_2)\}$

The Fourier transform of the ACF is called the **power spectrum**.

White noise and the noise to signal power ratio

If we assume **white noise** then the power ratio term in the denominator can be simplified:

$$R(\omega_1,\omega_2) = \frac{H^*(\omega_1,\omega_2)}{|H(\omega_1,\omega_2)|^2 + \frac{P_{NN}(\omega_1,\omega_2)}{P_{XX}(\omega_1,\omega_2)}} = R(\omega_1,\omega_2) = \frac{H^*(\omega_1,\omega_2)}{|H(\omega_1,\omega_2)|^2 + NSPR}$$

Assuming white noise

The term $NSPR = \frac{\sigma_N^2}{\sigma_X^2}$ is called noise to signal power ratio.

A variance of a signal can be known a priori (like we know that $\sigma_N^2 = 0.001$) or in MATLAB we can compute its value using `var(signal)`.

Exercise 5

Implement the function `restoration` in which:

- The function has at least 3 input parameters:
  - `y` is the degraded image,
  - `h` is the point spread function of the imaging system,
  - `method` is the chosen restoration approach
- The input `varargin` can have different sizes and its content depends on the chosen restoration method. The output is the restored `x_tilde` image.

It is advised to transform `y` and `h` into the frequency domain right at the beginning. In the upcoming operations `Y` and `H` have to be the same size. Use the built in `psf2otf` function to transform `h` and provide the target size as the second argument!

```
H = psf2otf(h, size(Y))
```

Exercise description continues on the next slide...

Exercise 5 – continued

Valid restoration methods are `'inverse'`, `'CLS'`, `'wiener'` and `'wiener-white'`.

- If `'inverse'` method is chosen `R` should be the matrix returned by your previously implemented `inverse_filter` function.
- If `'CLS'` method is chosen `R` should the matrix returned by your previously implemented `CLS_filter` function. In this case you can be sure that `varargin` contains two variables, `alpha` and `c` respectively.

  Warning! You have to transform `c` to `C` using the `psf2otf` function.

Exercise description continues on the next slide...

Exercise 5 – continued

- If `'wiener'` method is chosen `R` should the matrix returned by your previously implemented `wiener_filter` function. In this case you can be sure that `varargin` contains one variable which is the noise layer `n`. From this, you have to compute `P_NN` (see Slide 20) and from `y` compute `P_XX` too (just use `y` everywhere where `x` is supposed to be used). These power spectrum matrices must be matrices (with the same size as `n` and `y`)!

  Since we don't know the exact X, we compute P_{XX} from the signal Y.

Exercise description continues on the next slide...

Exercise 5 – continued

- If `'wiener-white'` method is chosen `R` should the matrix returned by your previously implemented `wiener_filter` function. In this case you can be sure that `varargin` contains one float which is the variance of the additive noise. Using `y` you have to estimate the image variance (see Slide 21) and call the wiener filter function with `(H, var_noise, var_image)` parameters!

Exercise description continues on the next slide...

Exercise 5 – continued

Use `if` statements and `strcmp` to check what type of method is required.

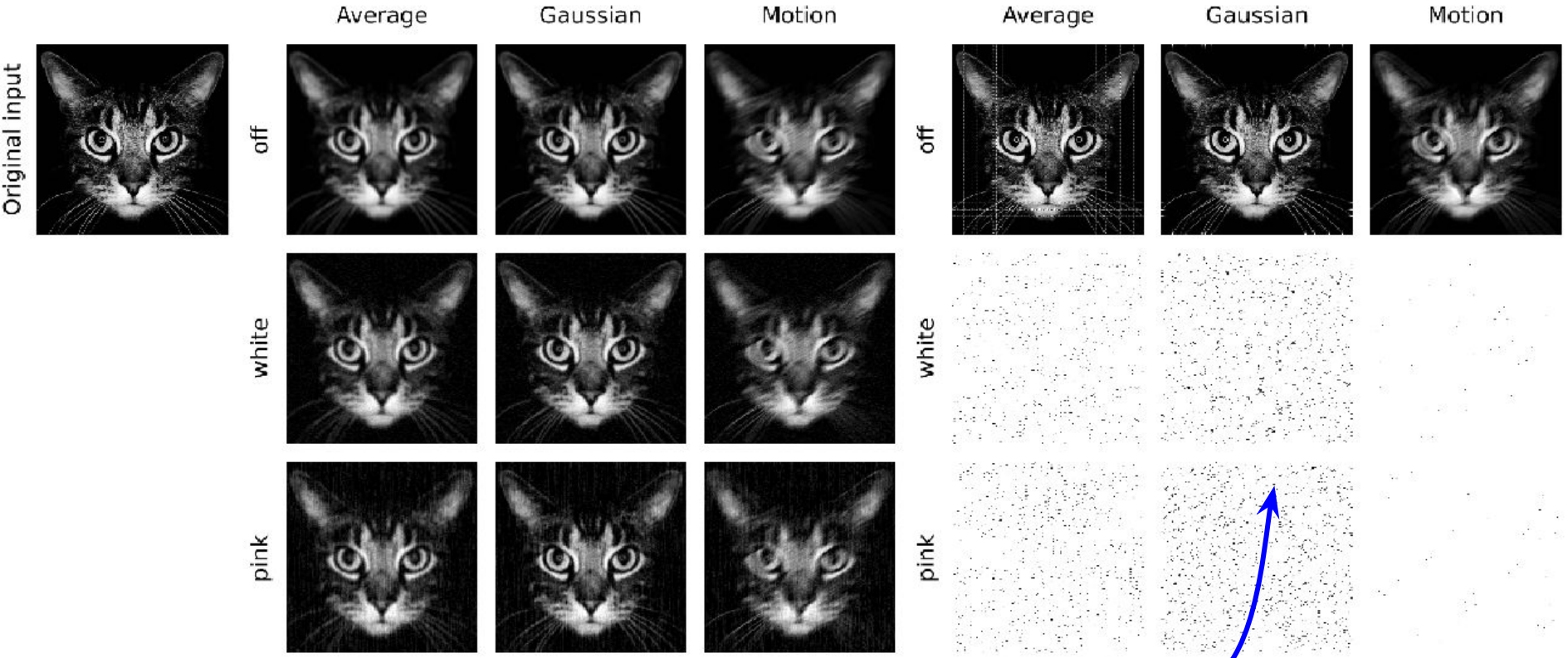
Raise an error (using `error()`) if the restoration method name is invalid.

After creating the appropriate `R` matrices, apply the restoration system to the input in the frequency domain. For this you have to take the Fourier transform of `y`, multiply `Y` with `R` elementwise and transform back the result to the spatial domain. Also, you have to take the absolute value of the inverse transform.

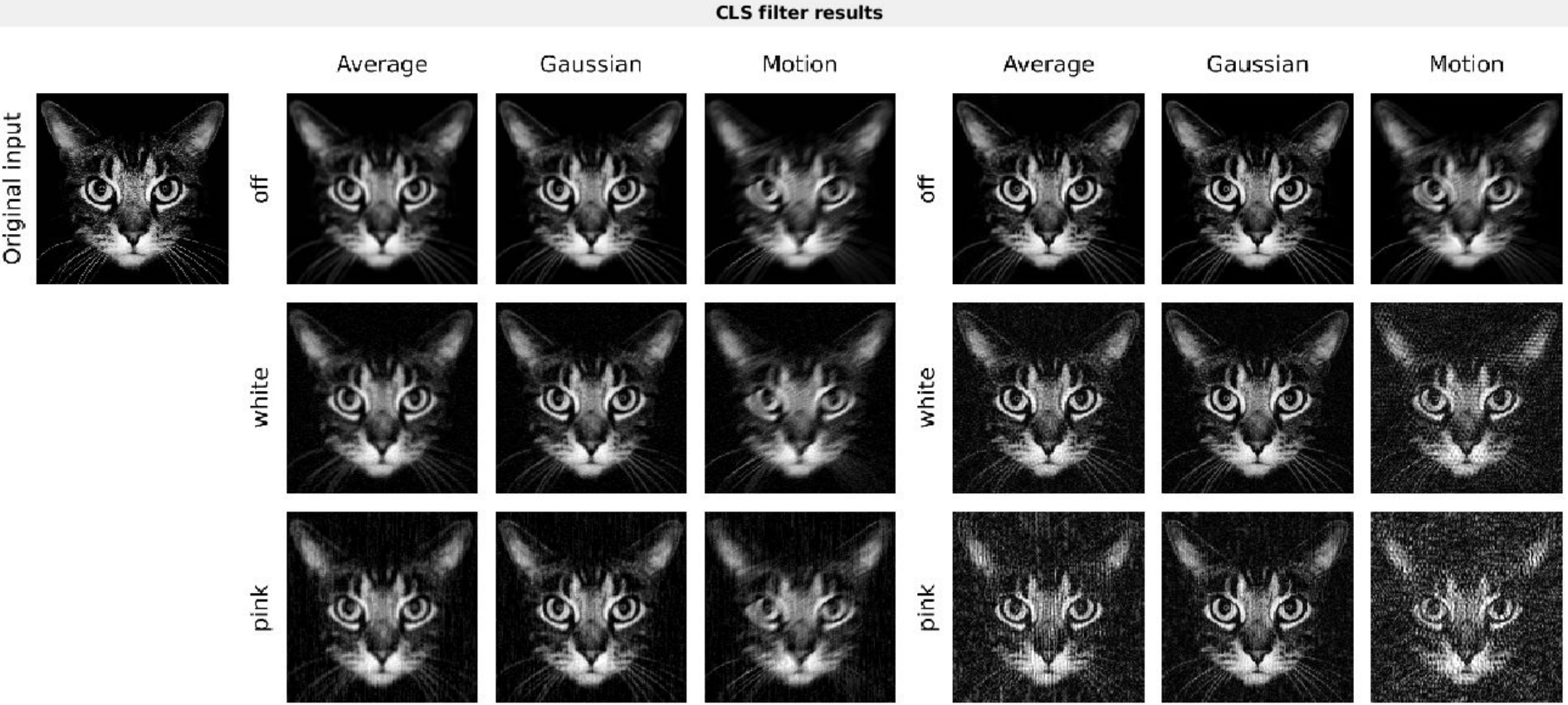
$$\tilde{x} = \left|\mathrm{IDFT}\left\{R \cdot \mathrm{DFT}\{y\}\right\}\right|$$

Execute `script2.m` and check your results!

Inverse filter results

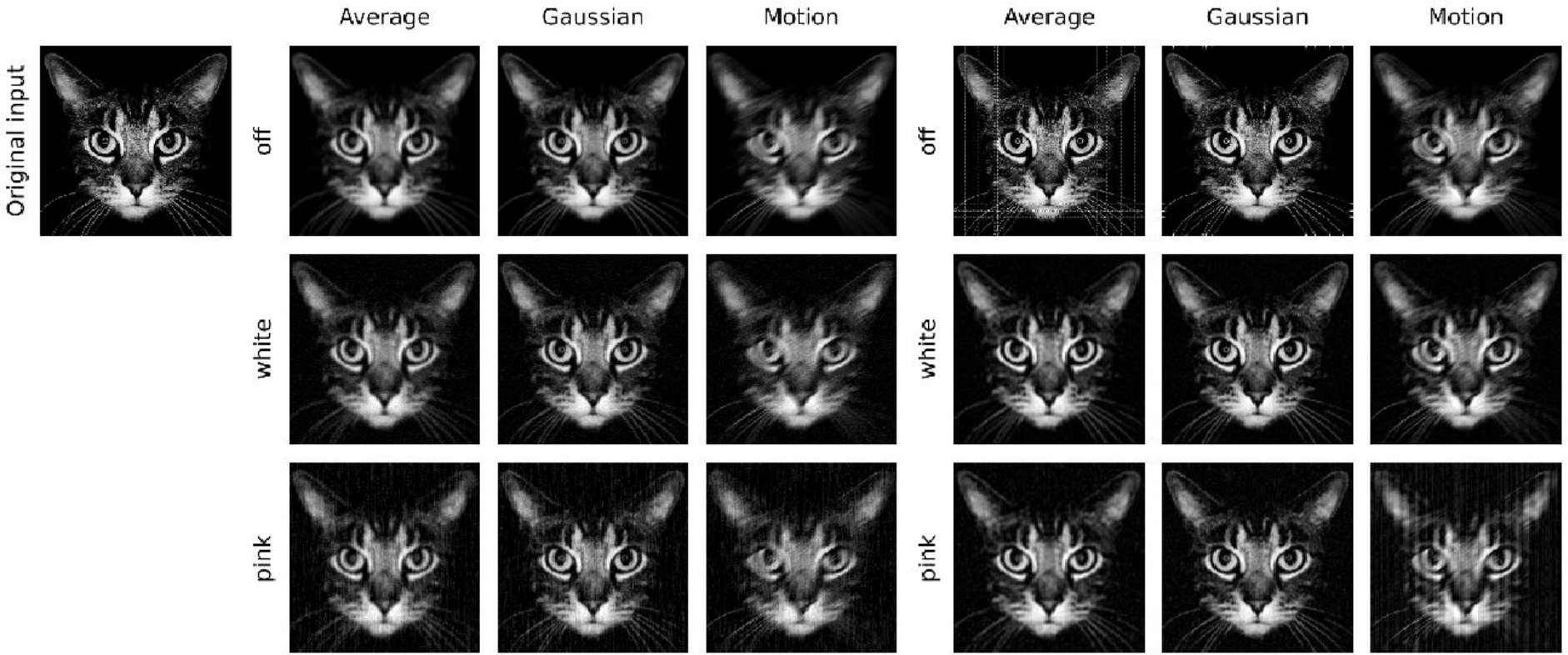


The basic inverse filter cannot handle the noise.

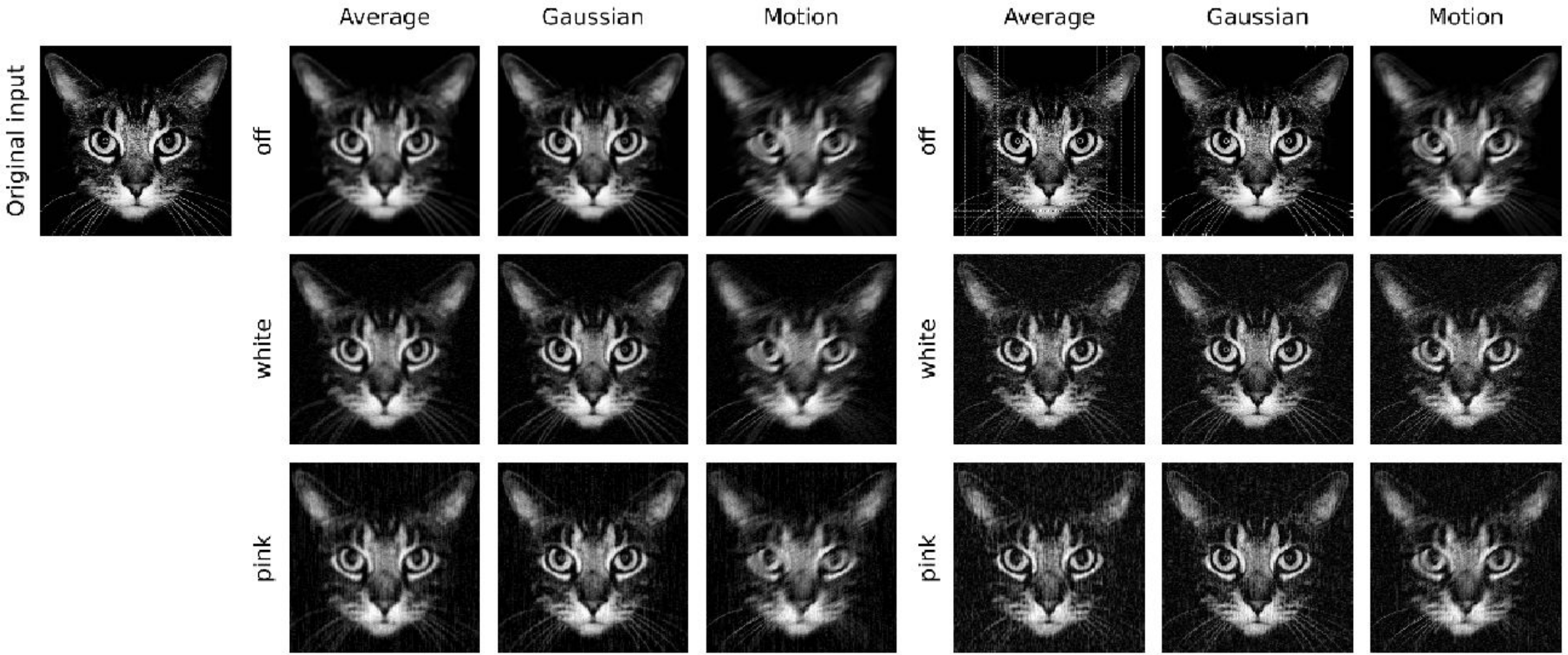


The noise filtering with CLS is definitely better but I had to adjust `alpha` and `c` manually.

Wiener filter results



Wiener filter gets rid of the noise quite well.



The pink noise removal is not so good if you assume white noise...

THE END