Topic 06 – Noise and filter

Open the image *plant-noise.tif*. The image contains a high level of noise. Zoom into the image. The background should be homogeneous, but it contains a random distribution of intensities. The same random distribution is added to the signal.

Excercise 6.1 – Noise

The noise can pose a problem for the analysis of the image, for example in the separation of the objects from the background. e might have to pre-process our image to reduce the disturbing effect of noise.

a) Open the image *plant-noise.tif* from the folder *06 - noise and filter*. Try to use the *Threshold-Adjuster* to separate the plant from the background. Is this possible?

Exercise 6.2 – Convolution Filter – Mean Filter

An evident way to smooth our image and to reduce the impact of the noise is to replace each pixel in the image by an average of the intensities in its neighborhood. This is called a mean filter in ImageJ. You can find it under *Process>Filters>Mean.*

a) Apply the mean-filter to the *plant-noise* image. Compare the histogram before and after application of the filter. What radius do you need to use, in order to be able to separate the plant from the background using the threshold-adjuster?
b) Imagine that your image is a white rectangle (intensity 255). How will the result of applying a mean-filter with radius 1 look like? Control your prediction by applying the filter to the image *rectangle.tif*.
c) Use the rectangle-image to compare the results of the command *Process>Smooth* with the results from the mean-filter with radius 1.
d) Use the convolver (*Process>Filter>Convolve*) to define a mean-filter. How does the kernel look like? On the plant-noise image, is the result of the convolver exactly the same as the result of the mean-filter command?

Exercise 6.3 - Convolution Filter – Gaussian Blur Filter

A Gaussian blur filter is similar to the mean filter, but instead of using uniform weights in the kernel, the weights are the values of a normal distribution, also called Gaussian distribution.

a) Open the file *Gaussian.txt*. Here you see the values of a normal distribution. How does it look like as an image? Open the file as an image by using *File>Import>Text Image*. Apply the ice lut. Now go to *Analyze>Surface Plot* or use the *Interactive 3D surface plot* plugin (*Plugins>3D*) to get an impression of the 3D form.
b) Apply the Gaussian blur filter with sigma 1.2 from *Process>Filters>Gaussian blur* to the rectangle image and compare the result with the result of the mean filter with radius 3.
c) Apply a Gaussian blur filter with a sigma of 6 and a Mean filter of radius 15 to the image *roots.tif* and compare the results.

d) Open the plant-noise image. Open the convolution tool and load the *Gaussian.txt* into it. Run the convolution filter on the image.

Exercise 6.4 – Edge enhancing convolution filter - Sobel and Prewitt filter

Other kernels can be used to create edge enhancing filters.

a) The Prewitt filter is defined by the kernels:
$k_x = \begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix}$ and $k_y = \begin{matrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{matrix}$. If D_x and D_y are the results of the convolution of k_x and k_y with the input image then you have to calculate $\sqrt{D_x^2 + D_y^2}$ to apply the Prewitt filter. Apply the Prewitt filter to the image rectangle.tif. Create a composite image of the input image and the result (*Image>Color>Merge Channels*) Hints:

- convert the image to 32-bit first, make a copy and use the convolution-tool with the two kernels
- you find the square and square-root operations under *Process>Math*
- Use the image calculator (*Process>Image Calculator...*) to add two images

b) Apply the Prewitt filter to the image *roots.tif*. Don't forget to convert it to 32-bits. Create an overlay of the input image and the filtered image.

c) Another edge enhancing filter is the Sobel filter. It works in the same way but with the kernels:
$k_x = \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$ and $k_y = \begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$. Apply the Sobel-filter to the image *rectangle.tif* and compare the result with the result of the command *Process>Find Edges*

d) High-pass or *Uni Crisp* filter can be used to apply a sharpening effect to an image. Convolve the image *ophrys.tif* with the kernel $\begin{matrix} -1 & -1 & -1 \\ -1 & 12 & -1 \\ -1 & -1 & -1 \end{matrix}$ and compare the result with the result of the command *Process>Sharpen.*

e) A Laplacian of Gaussian (LoG) filter also called Mexican Hat Filter applies a gaussian smoothing filter and an edge enhancing filter in one step. It can be used to detect spots. Possible discrete approximations for the laplacian kernel are:
$\begin{matrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{matrix}$ or $\begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix}$. You can apply a LoG filter by first applying a gaussian filter and then convolving with one of the above kernels. However the gaussian and laplacian can be combined in one single kernel. Install the LoG plugin from http://bigwww.epfl.ch/sage/soft/LoG3D/. Try to threshold the bright spots on the image. Run the LoG plugin on the image. While applying it create the kernel as an image and have a look at it. Try thresholding the spots after application of the filter.

Exercise 6.4 – Ranking filter – Median filter

Rank filters are another class of filters. Again we look at a neighborhood of radius r for each pixel. But this time the intensity values are sorted and the pixel is replaced with the value in a specific position. If we use the middle position we get a median filter.

a) Open the image *plant-sp-noise.tif*. Zoom in to the image. You see that in this case the noise consists of the addition of white and black pixels to the image. This kind of noise is called salt-and-pepper noise. Apply a median filter of radius 3 from *Process>Filters>Median* to the image. Compare the result with the result from the mean filter.
b) Compare the effect of the median filter with radius 1 and the command *Process>Noise>Despeckle*.
c) In *Process>Noise>Remove Outliers*, you find a selective median filter, that replaces a pixel by the median in the neighborhood if the value of the pixel differs more than a threshold value from the median. Use it to remove the dark and light components of the salt and pepper noise separately.
d) In *Process>Noise>Remove Outliers*, you find a selective median filter, that replaces a pixel by the median in the neighborhood if the value of the pixel differs more than a threshold value from the median. Use it to remove the dark and light components of the salt and pepper noise separately.
e) The signal-noise ratio is a measure of how disturbing the noise is. It can be calculated in the following way:
$\frac{IntDen(I^2)}{IntDen[(I-M)^2]}$ where I is the ideal image not affected by noise, M the measured, noisy image and IntDen the integrated intensity, i.e. the sum of all intensity values in the image. With the help of the macro recorder, create a macro that calculates the snr for the example image boats from the menu *File>Open Samples*. Create the measured image M by adding specified noise with sigma = 15 to the input image. Use the macro to calculate the snr after applying a gaussian-blur filter with sigma 1.2, a mean filter with radius 3 and a median filter with radius 3. Compare the results. Do it with other example images, as well.

Exercise 6.5 – Ranking filter – Min and Max filter – Grey-morphology

Taking the minimum or maximum value instead of the median gives the min- and max-filter. The min filter increases dark regions, the max filter bright regions. We can use them to define a complete grey-morphology, with the only restriction that the structuring element is fixed to a square of radius r. Some of the operations are:

- min – erode – enlarge dark regions
- max – dilate – enlarge bright regions
- open – dilate(erode(n)) – remove light features smaller than r
- close – erode(dilate(n)) – remove dark features smaller than r
- sequential filter – open(close(n)) – remove bright and dark features smaller than r
- white top hat – I-opening(I) – extract bright features

- black top hat – closing(I)-I – extract dark features
- …

a) Apply the min and max filter to the image *Lena.tif* try the values 1, 10 and 20 for the radius.

b) Create a toolset *simple grey morphology* that has a button for each of the operations erode, dilate, open, close, sequential filter, white top hat and black top hat. A right click on the first button should open a dialog in which the user can set the radius that will be used by all the operations. Run the operations on the image lena.tif and clown.jpg with different values for the radius. Hints:

- Use the macro editor to write a new toolset (*Plugins>New>Macro*).
- You can open an existing toolset to have a look at an example by holding down shift and switching to the toolset with the >> button.
- A macro has the form
```
macro "Erode Action Tool- C000T4b12e" {
 …
}
```
The string `C000T4b12e` defines the image on the button. In this case it draws the letter e in black.
- Declare a variable for the radius of the min and max filter at the beginning
```
var radius = 1;
```
To add an option that allows to set the radius to a button you can use for example:
```
macro 'Erode Action Tool Options' {
 radius = getNumber("radius:",radius);
}
```
- You can declare functions that you can call from a macro:
```
function erode() {
 run("Minimum...", "radius=" + radius);
}
```
- You can use the macro-recorder if you don't know how to write a specific command in the macro language.

Exercise 6.6 – Filtering in the Frequency Domain

The Fourier transform changes the coordinate system from the spacial coordinates to the coordinates angle and frequency. The basis is that any signal can be approximated by a sum of harmonic functions (sin and cos) of different amplitude and phase. The result is complex numbered, so we really transform one spacial image into two result images: the phase image (imaginary part) and the powerspectrum (real part). The part the more interesting is the powerspectrum.

According to the convolution theorem, a point-wise multiplication in the frequency domain corresponds to a convolution in the spacial-domain. Besides the transform and the inverse transform

can efficiently be calculated. So we can use it to more efficiently apply convolution filters that do not have a simple kernel in the spacial-domain.

Possible applications are:

- low-pass filter – smoothing – noise suppression
- high-pass filter – edge detection
- band-pass filter – detect structures with a given frequency
- correlation, template matching, stitching
- deconvolution (inverse filtering) (see 4.5 d) and e))

ImageJ allows to apply a Fourier transform under *Process>FFT>FFT*. As a result the powerspectrum is displayed. This is a special image on which you can define a filter by making regions white or black. Frequencies under the white mask will be passed, those under the black mask will be suppressed when you calculate the inverse transformation (*Process>FFT>Inverse FFT*).

a) Open the image *pierre.tif*. Calculate the Fourier transform. Do you see the two clusters on in the upper left half and one in the lower right? Draw a white mask on them and calculate the inverse transform. Is there an object in the image that corresponds to the frequencies that passed?

b) Use the Fourier transform to suppress the noise in the image *plant-noise.tif*. Which frequencies do you have to suppress?

c) Use the Fourier transform to detect the edges in the image *ophrys.tif*. Which frequencies do you have to suppress?

d) The image *lines.tif* consists of vertical white lines that have a width of one pixel and a distance of 4 pixel. How will the powerspectrum of the Fourier transform of the image look like? Verify your answer by creating it.

e) Load the image *text.tif* and the image *template.tif*. Use the command *correlate* from *Process>FFT>FD Math* to calculate the correlation between the text and the template. Transform the resulting (spacial) image to 8-bit, move the minimum threshold value so the right until only single points remain and apply the threshold. Now use *Process>Find Maxima* to create a point selection from the points and transfer the point selection onto the text image, to see which points gave the maximum response to the template.