

# Lab Instructions - session 4

## Noise and Filtering

Adding Gaussian noise to image.

File: `add_gaussian_noise.py`

```
import numpy as np
import cv2

I = cv2.imread('isfahan.jpg', cv2.IMREAD_GRAYSCALE);

# convert I to floating point from unsigned integer
# Note: For displaying floating point images the maximum
# intensity has to be 1 instead of 255
I = I.astype(np.float) / 255

# create the noise image
sigma = 0.04 # notice maximum intensity is 1
N = np.random.randn(*I.shape) * sigma

# add noise to the original image
J = I+N; # or use cv2.add(I,N);

cv2.imshow('original',I)
cv2.waitKey(0) # press any key to exit

cv2.imshow('noisy image',J)
cv2.waitKey(0) # press any key to exit

cv2.destroyAllWindows()
```

- What does the line `I = I.astype(np.float) / 255` do?
- An asterisk "\*" before an argument in a python function call, gives elements of the argument (which is typically a tuple) as argument to the function. In the above, if `I.shape = (200,300)`, then `np.random.randn(*I.shape)` is the same thing as `np.random.randn(I.shape[0], I.shape[1])`. Similarly, if `I.shape = (200,300,3)` for a color image, then `np.random.randn(*I.shape)` is equivalent to `np.random.randn(I.shape[0], I.shape[1], I.shape[2])`.
- Try different values of **sigma** and see the result. What is the effect of small/large sigma on noise?
- Change `cv2.IMREAD_GRAYSCALE` to `cv2.IMREAD_COLOR` in `imread` (or remove this argument) and see the result.

## Task 1:

We want to simulate white noise (**snow noise** or **Barfak** in Persian!) in the old analog TVs. We read an image, and then, in every iteration of a loop, we add a **different** randomly generated Gaussian noise to it. We show the image at around **30 frames per second** (Hence the command `cv2.waitKey(33)`). Notice that you need to create a new noise image at every new frame (in most cases with the same sigma). Your program must increase or decrease the intensity of noise when the user presses the keys 'u' or 'd' respectively. Read the file `snow_noise.py` and change it to create this demo. **Notice that sigma should never get negative.**

File: `snow_noise.py`

```
import numpy as np
import cv2

I = cv2.imread('isfahan.jpg', cv2.IMREAD_GRAYSCALE);
I = I.astype(np.float) / 255

sigma = 0.04 # initial standard deviation of noise

while True:

    J = I; # change this line so J is the noisy image

    cv2.imshow('snow noise', J)

    # press any key to exit
    key = cv2.waitKey(33)
    if key & 0xFF == ord('u'): # if 'u' is pressed
        pass # increase noise
    elif key & 0xFF == ord('d'): # if 'd' is pressed
        pass # decrease noise
    elif key & 0xFF == ord('q'): # if 'q' is pressed then
        break # quit

cv2.destroyAllWindows()
```

## Image Smoothing/Blurring

Smoothing an image with a box kernel

file: `blur_box.py`

```
import numpy as np
import cv2

I = cv2.imread('isfahan.jpg').astype(np.float64) / 255;

# display the original image
cv2.imshow('original', I)
cv2.waitKey()

# creating a box filter
m = 7 # choose filter size

# create an m by m box filter
F = np.ones((m,m), np.float64) / (m*m)
print(F)

# Now, filter the image
J = cv2.filter2D(I, -1, F)
cv2.imshow('blurred', J)
cv2.waitKey()

cv2.destroyAllWindows()
```

- Alter the value of `m` and see what happens.
- Why the division by `(m*m)` in `F = np.ones((m,m), np.float64) / (m*m)`?
- You can also apply a box filter to an image using the [cv2.boxFilter](#) function.

## Smoothing with a Gaussian Kernel

Here, we first create a one-dimensional Gaussian kernel. Then make a two-dimensional Gaussian kernel out of the 1D kernel, and apply the 2D kernel to the image.

file: **blur\_gaussian.py**

```
import numpy as np
import cv2

I = cv2.imread('isfahan.jpg').astype(np.float64) / 255;

m = 13; # we will create an m by m filter

# create a 1D Gaussian filter
Fg = cv2.getGaussianKernel(m, sigma=-1);
# by setting sigma=-1, the value of sigma is computed
# automatically as: sigma = 0.3*((ksize-1)*0.5 - 1) + 0.8

print(Fg)
print(Fg.shape) # Fg is 1-dimensional (m by 1)

exit() # delete this to continue

# Now we create a 2D filter
# We use matrix multiplication to create an m by m 2D filter
# out of "m by 1" and "1 by m" 1D filters, which in this case happens
# to be the same thing as the correlation between 1D filters
Fg = Fg.dot(Fg.T) # an "m by 1" matrix multiplied by a "1 by m" matrix

print(Fg)
print(Fg.shape)

exit() # delete this to continue

# filter the image with the Gaussian filter
Jg = cv2.filter2D(I,-1, Fg)

cv2.imshow('original',I)
cv2.waitKey()

cv2.imshow('blurred_Gaussian',Jg)
cv2.waitKey()

cv2.destroyAllWindows()
```

- Alter the value of m and see what happens. Notice that altering m will alter filter  $\sigma = 0.3*((m-1)*0.5 - 1) + 0.8$ . You can also give sigma explicitly.
- You can simply [gaussianBlur](#) function instead. Also, look [here](#).

## Task 2:

This is an extension to **Task 1**. This time we also filter the noisy image using the box and Gaussian filters. You need to complete the file **noise\_filter\_demo.py**. Your program must have the following functionalities:

- **press the 'b' key:** use box filter
- **press the 'g' key:** use Gaussian filter
- **press the '+' key:** increase filter size **m**
- **press the '-' key:** decrease filter size **m**
- **press the 'u' key:** increase noise intensity
- **press the 'd' key:** decrease noise intensity
- **press the 'q' key:** quit the program

File: **noise\_filter\_demo.py**

```
import numpy as np
import cv2

I = cv2.imread('isfahan.jpg').astype(np.float64) / 255;

noise_sigma = 0.04 # initial standard deviation of noise

m = 1; # initial filter size,
# with m = 1 the input image will not change
filter = 'b' # box filter

while True:
    if filter == 'b':
        # filter with a box filter
        F = np.ones((m,m), np.float64)/(m*m)

    elif filter == 'g':
        # filter with a Gaussian filter
        pass

    # add noise to the image
    J = I + N;

    # filtered image
    K = cv2.filter2D(J, -1, F);

    cv2.imshow('img', K)
    key = cv2.waitKey(30) & 0xFF

    if key == ord('b'):
        filter = 'b' # box filter
```

```
    print('Box filter')
elif key == ord('g'):
    filter = 'g' # filter with a Gaussian filter
    print('Gaussian filter')

elif key == ord('+'):
    # increase m
    m = m + 2
    print('m=', m)
elif key == ord('-'):
    # decrease m
    if m >= 3:
        m = m - 2
    print('m=', m)

elif key == ord('u'):
    # increase noise
    pass

elif key == ord('d'):
    # decrease noise
    pass

elif key == ord('q'):
    break # quit

cv2.destroyAllWindows()
```

- Change the noise intensity. In each case try to find the optimal filter size **m**.
- Compare the Gaussian filter with the box filter. Which one performs better?

## References

1. [https://docs.opencv.org/3.0-beta/doc/py\\_tutorials/py\\_imgproc/py\\_filtering/py\\_filtering.html#filtering](https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_filtering/py_filtering.html#filtering)