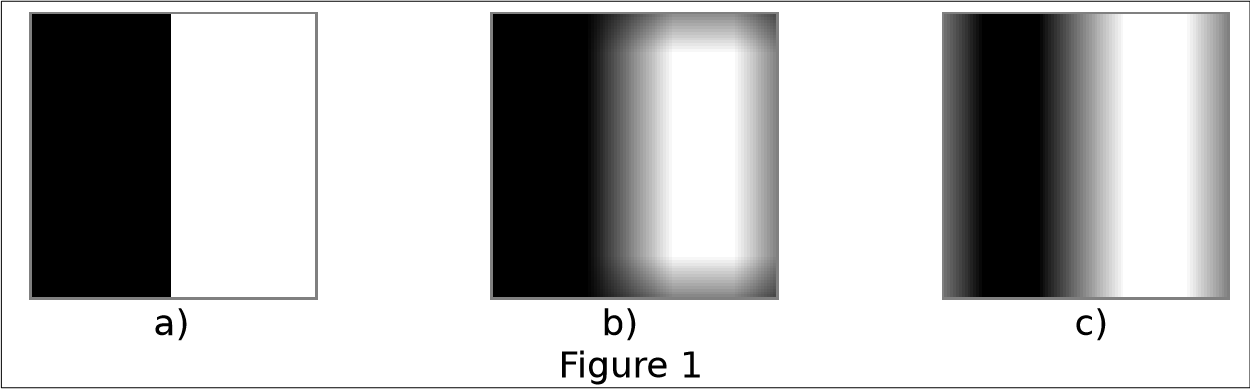
Group Q: Fuhrmann, Robert (364623)  
Tkaczyk, Tomasz Henryk (368998)

# Exercise 3

The usm and threshold functions are working pretty good, till a filter kernel size of 25. Above this size, there seems to be a problem with the image range. The other functions do well, so far.

For the images in the submission we used a threshold of 10 and a scale of 2.

## Theory



A moving average filter was applied to the image in Figure 1a). Figure 1b) shows the result, if the convolution is carried out in spatial domain, Figure 1c) if the convolution is carried out as multiplication in frequency domain.

1. Explain which assumptions lead to the “unexpected” border values in each image and why they are different for both methods.

In b) we assume, that every pixel next to our image has the value zero. This leads to a darker value of the pixels at the border. The view could be changed, by using a different type of boarder handling in spatial domain.

In c) there was used a discrete fourier transformation which is described as a periodic function. The image is repeated to all sides, thus pixels on borders have an influence on the opposite border of the image.

1. ii) What steps are necessary for the convolution in spatial domain to produce the result in Fig. 1c)?

For the result of Fig. 1c) in spatial domain, we need to use the wrapping method for border-handling. This means, when the filter kernel step out of our image at the right side, we use pixels from the very left side of the image for convolution, an converse.

1. iii) What steps are necessary for the convolution by multiplication in frequency domain to produce the result in Fig. 1b)?

To achieve the result in Fig 1b) we could attach a black border to the image before convolution and remove it after the execution. One way to do this is by using the copyMakeBorder() funkction of OpenCV.

## Time behaviour of convolution

One think we can see is, that there Is no big impact from the size of the filter kernel to the time needed for convolution. The effect is stronger in frequency domain. The reason for this behaviour might be the very simple border handling in our program. Methods like wrapping or extending the image would claim much more time, depending on the filter size.

x = number of pixels, y = size of filter, z seconds for execution