WS 2019/20	Exercises Digital Image Processing I	Solution of Exercise No. 13

Function Padding

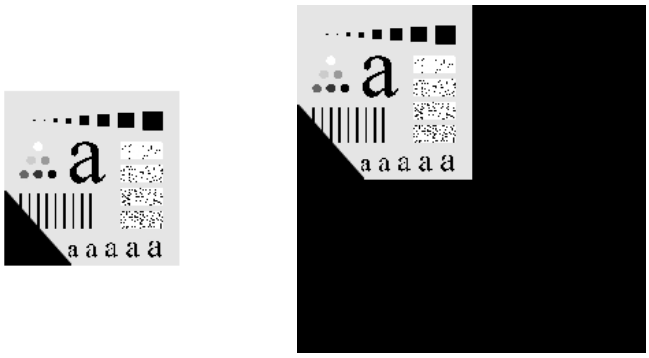
Exercise goal:

Allgemein

- Function Padding
- Implement filter and folding function
- Filtering and folding with komplexwertigen images with and without Function Padding
- Filtering in the frequency domain

Task: *Write a function which fills an image with zeros according to the rules of Function Padding!*

The Function Padding expands the image by $M-1$ and $N-1$ in the x and y direction and is filled with zeros, where $M \times N$ is the size of the filter to be used for foling and/or filtering the image. It is expanded to the right and downward.



(a) (b)

Abbildung 1: Function Padding with an image and filter size of 128×128, (a) normal, (b) expanded to 255×255 through Function Padding

Determination of interface:

```
// Fills an image with zeros (Function Padding). The input image is delivered to
// <input> and the filled image is saved in <padded> .
void functionPadding (const ComplexImage& input, ComplexImage& padded)
```

Implementation:

```
//
// (11.1a) Fuellt ein Bild mit Nullen auf (Function Padding). Das Eingabebild wird in
//         <input> uebergeben und das aufgefuellte Bild in <padded> gespeichert.
//
void functionPadding (const ComplexImage& input, ComplexImage& padded)
{
    int       width   = padded.getWidth();
    int       height  = padded.getHeight();
    Complex* pdata   = padded.getData();

    int       iwidth  = input.getWidth();
    int       iheight = input.getHeight();
    Complex* idata   = input.getData();

    for (int y=0; y<height; y++)
    {
        for (int x=0; x<width; x++)
        {
            if ((x<iwidth) && (y<iheight))
            {
                pdata[y*width+x] = idata[y*iwidth+x];
            }
            else
            {
                pdata[y*width+x].re = 0.0;
                pdata[y*width+x].im = 0.0;
            }
        }
    }
}
```

Solution for task 13.1b

Task: *Write two functions, each of which carry out one folding and on filtering of komplex-wertiger images with random filter masks.*

The difference between filtering and folding is (in terms of signal theory) that during filtering a definite property of an input signal (i.e. the input image) is extracted and emphasized. Folding generates a new signal (new image)by superimposing two signals (old image and filter). Folding is a type of specific filtering, but not all filterings are foldings.

1. Complex filtering functions:

During the lecture the filtering function (compare `filter(...)` function from exercise 6.1d) is defined as:

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t) \tag{1}$$

with f being the input image, w the filter of size $M \times N$, $M = 2a+1$ and $N = 2b+1$, and g the resulting image. It has to be kept in mind that the starting point of the filter w is at the center pixel set at the position $f(x,y)$ of the image. Unlike in folding, the filter is not reflected and the starting point of the filter is the center pixel.

2. Complex folding function:

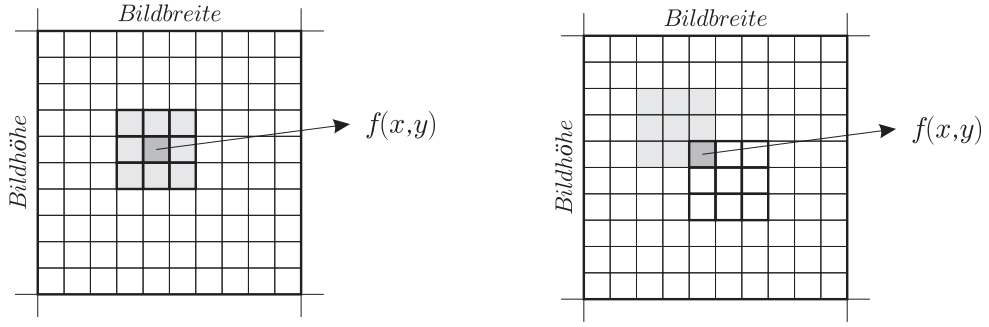
The folding function described during the lecture is to be implemented. It is defined for two functions $f(x,y)$ and $h(x,y)$ of size $M \times N$ as:

$$g(x,y) = f(x,y) * h(x,y) \tag{2}$$

$$g(x,y) = \frac{1}{MN} \sum_{s=0}^{M-1} \sum_{t=0}^{N-1} h(s,t) f(x-s, y-t) \tag{3}$$

with f being the input image, h ther filter and g the resulting image. $\frac{1}{MN}$ is a constant factor and corresponds to the weighting across all elements. This factor is omitted in order to allow a comparison with the filterning. The starting point of the filer h is at the position $h(0,0)$ and at the image position $f(x,y)$.

Image 2 graphically displays the differences between filtering and folding (here with a 3×3 filter as an example). With a $M \times N$ filter the filter exceeds the edges of the image and has to be handled accordingly. The light gray area is the observed part of the image, whereas the filter is displayed in bold. The center pixel is at the position $f(x,y)$ which is to be observed. Effects on the images are shown in exercise 9.1c.



(a) (b)

Abbildung 2: (a) Filterung, (b) Faltung

Determination of the interface (Complex filtering):

```
// Complexe filtering. As convolve(...) with the difference that
// filtering and not folding is carried out.
void filter (ComplexImage& input_image, ComplexImage& output_image,
             ComplexImage& mask)
```

Implementation:

```
//
// (11.1b) Komplexe Filterung (vgl. Filterung im Ortsraum, Aufgabe 6.1d). Wie convolve(...)
//         mit dem Unterschied, dass eine Filterung und keine Faltung durchgefuehrt
//         wird (Praktisch besteht der Unterschied darin, dass die Maske nicht
//         gespiegelt wird und ausserdem der Aufsatzpunkt nicht der Position (0,0)
//         sondern der Maskenmitte entspricht).
//
void filter (ComplexImage& input_image, ComplexImage& output_image, ComplexImage& mask)
{
    int       mwidth   = mask.getWidth();
    int       mheight  = mask.getHeight();
    Complex* mdata    = mask.getData();

    int       mwidth2  = mwidth / 2;
    int       mheight2 = mheight / 2;

    int       iwidth   = input_image.getWidth();
    int       iheight  = input_image.getHeight();
    Complex* idata    = input_image.getData();
    Complex* odata    = output_image.getData();

    Complex sum;

    for (int y=0; y<iheight; y++)
    {
        for (int x=0; x<iwidth; x++)
        {
            sum.re = 0.0;
            sum.im = 0.0;
            for (int t=0; t<mheight; t++)
            {
                for (int s=0; s<mwidth; s++)
                {
                    int y2 = mod(y+(t-mheight2), iheight);
                    int x2 = mod(x+(s-mwidth2), iwidth);

                    sum += mdata[t*mwidth+s] * idata[y2*iwidth+x2];
                }
            }
            odata[y*iwidth+x] = sum;
        }
    }
}
```

determination of an interfache (Complex folding):

```
// Complee folding. The komplexwertige input image <input_image>
// is folded using the komplexwertigen filter mask <mask> . The komplexwertige
// result is written into <output_image> .
void convolve (ComplexImage& input_image, ComplexImage& output_image,
               ComplexImage& mask)
```

Implementation:

```
//
// (11.1b) Komplexe Faltung. Es wird das komplexwertige Eingangsbild <input_image>
//         mit der komplexwertigen Filtermaske <mask> gefaltet. Das komplexwertige
//         Ergebnis wird nach <output_image> geschrieben.
//
void convolve (ComplexImage& input_image, ComplexImage& output_image, ComplexImage& mask)
{
    int       mwidth  = mask.getWidth();
    int       mheight = mask.getHeight();
    Complex* mdata   = mask.getData();

    int       iwidth  = input_image.getWidth();
    int       iheight = input_image.getHeight();
    Complex* idata   = input_image.getData();
    Complex* odata   = output_image.getData();

    Complex  sum;

    for (int y=0; y<iheight; y++)
    {
        for (int x=0; x<iwidth; x++)
        {
            sum.re = 0.0;
            sum.im = 0.0;
            for (int t=0; t<mheight; t++)
            {
                for (int s=0; s<mwidth; s++)
                {
                    int y2 = mod(y-t, iheight);
                    int x2 = mod(x-s, iwidth);

                    sum += mdata[t*mwidth+s] * idata[y2*iwidth+x2];
                }
            }
            odata[y*iwidth+x] = sum;
        }
    }
}
```

Solution for task 13.1c

Task: *Utilize the function from exercises a) and b), to fold and filter an image* mit *and* ohne Function Padding *using the filter masks of the filter in the position space in exercise 10.1c)!*

The following parameters were selected for the ideal, the Butterworth and the Gaussian low-pass filter: Cut-Off frequency $D_0 = 15, n = 2, \sigma = 10$. The figures 3, 4 and 5 each show folding and filtering of the filter using image *char2c.bmp*.

Eingabe:

(1) (2)

(1) input image *char2c.bmp*, (2) Real- and imagiary part of the ideal low-pass filter

Output (folding):

(a) (b) (c)

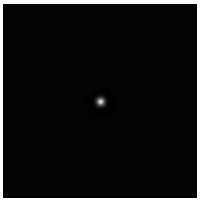
Output (filtering):

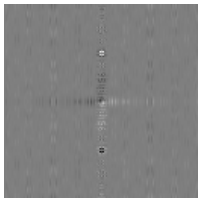
(a) (b) (c)

Abbildung 3: (a) wothout Function Padding. (b) with Function Padding. (c) filtering resulting with Function Padding (excerpt from (b) in the size of the original image).

Input:

(1)





(2)

(1) Input image *char2c.bmp*, (2) Real- and imaginary part of the Butterworth low-pass filter

Output (folding):

(a)

(b)

(c)

Output (filtering):

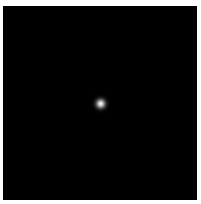
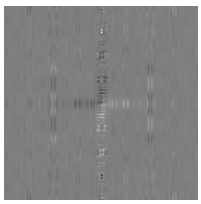
(a)

(b)

(c)

Abbildung 4: (a) without function padding. (b) with function padding. (c) Result of filtering with function padding (excert from (b) in the size of the original image).

Eingabe:

(1) (2)

(1) Eingabebild *char2c.bmp*, (2) Real- and imaginary part of the Gaussian low-pass filter

Ausgabe (Faltung):



(a) (b) (c)

Output (filtering):



(a) (b) (c)

Abbildung 5: (a) without function padding. (b) with function padding. (c) Result of filtering with function padding (excerpt from (b) in the size of the original image).