WS 2019/20	Exercises Digital Image Processing	Solution Exercise No. 6

Solution for task 6.1a

Task: *Implements a mean filter with a neighborhood size of* 3×3 *pixels. Give the formula for calculating the new value* $g(x, y)$ *of a pixel at* $f(x, y)$*!*

A mean filter replaces each pixel of an image by the avarage value of its neighborhood. The neighborhood shall consist of the pixel itself and its 8 direct neighbors (3x3 Pixel).

$$g(x,y) = \frac{f(x-1,y-1)+f(x,y-1)+f(x+1,y-1)+f(x-1,y)+f(x,y)+f(x+1,y)+f(x-1,y+1)+f(x,y+1)+f(x+1,y+1)}{9}$$

Solution for task 6.1b

Task: *What is the function of the filter?*

The function of the mean filter is smoothing. Smoothing can reduce image noise while keeping rougher structures. Related to the frequency spectrum of an image, smoothing equates to a "Low-Pass" filter. Contrary to a naive estimation [mean filters] are not real"Low-Pass" filter.

Solution for task 6.1c+d

Task: *In which positions can you not use the formula? Which method of boundary treatment makes sense to you for the mean filter?*

The boundary of an image have to be treated specially.

Stratigies for Boundary treatment:

- No computations for *border pixels*
  - Dye black or white.
    disadvantage: The image will become smaller with each filter operation because of the lost of the image information at borders.
  - Leave the gray values of the original image without any change.
- Computing *border pixels*
  - use adapted masks
  - use original mask
    → Pixel values from *outside* the image are required
    Ways to approximate the values of missing pixels outside the image:
    * Assume constant value (often 0)

* Approximate through boundary pixels
* Assume image to be cyclical (neighboring pixel of the left edges is the right edge..etc.)

The easiest strategy for boundary treatment was selected for the implementation in exercise 5.2e. is the "no computation for edge pixels" .

Solution for task 6.1e

Task: *Implement and test an algorithm for calculating the mean filter!*

Determination of interface:

```
// Computation of mean filter for <input>. The result
// is written in <output>.
void averaging3 (GrayImage& input, GrayImage& output);
```

Implementation:

```
//
// (5.2e)
//
void averaging3 (GrayImage& input, GrayImage& output)
{
    int     width  = input.getWidth();
    int     height = input.getHeight();

    float* idata  = input.getData();
    float* odata  = output.getData();

    float  sum;
    for (int y=1; y<height-1; y++)
    {
        for (int x=1; x<width-1; x++)
        {
            sum = 0;
            for (int y1=y-1; y1<=y+1; y1++)
            {
                for (int x1=x-1; x1<=x+1; x1++)
                {
                    sum += idata[y1*width+x1];
                }
            }
            sum = sum / 9;

            odata[y*width+x] = sum;
        }
    }
}
```

Solution for task 6.1f

Task: *How many arithmetic operations (additions / subtractions, multiplications / divisions) does the algorithm require for an image with height height and width width?*

The computation here is carried out without taking into consideration the applied boundary treatment!
The algorithm runs through the y-loop *height* times, since the image has *height* lines. During each run of a y-loop the algorithm runs through the x-loop *width* times. The computation of a new pixel value is thus done by $height * width$ times. This obviously corresponds to the number of pixels. If the computation of the pixel index is neglected ($y * height + x$) 9 Additions and 1 Division are carried out.

Thus the algorithm requires:

$9 * height * width$	Additions/Subtractions
$height * width$	Multiplications/Divisions.

Main Program:

```
{
    /* ********************************** */
    /* Exercise 05                        */
    /* ********************************** */

    string filename;
    cout << "Filename: ";
    cin >> filename;

    GrayImage image;
    image.load(filename);
    GrayImage result(image.getWidth(), image.getHeight());

    cout << "Chose a Function" << endl;
    cout << "----------------" << endl;
    cout << "1 - Scaling" << endl;
    cout << "2 - Gamma correction" << endl;
    cout << "3 - 3 x 3 Average filter" << endl;
    int fkt;
    cin >> input;

    switch (input)
    {
        case 1:
            scale(image, result);
            break;
        case 2:
            cout << "Gamma: " << endl;
            float g;
            cin >> g;

            gamma(image, result, g);
            break;
        case 3:
            averaging3(image, result);
            break;
    }

    image.show();
    result.show();

    cout << "FINISHED.\n";
    return 0;
}
```