WS 2019/20	**Exercises Digital Image Processing**	**Solution of Exercise No. 4**

Periodic Patterns - Frequencies in Digital Image Processing

Goal of the Exercise:

- Understanding of spatial frequencies (described by periodic patterns)
- Description of periodic patterns by the wavenumber vector
- Calculation of periodic patterns
- Sampling and sampling theorem
- Failure to comply with the sampling theorem

Solution for task 4.1a

Task: *Describe the following periodic pattern using the wavenumber vector!*

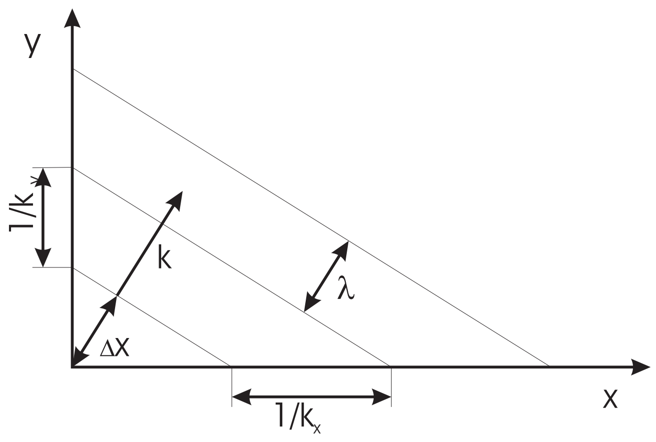

Figure 1: *Description of the periodic 2D pattern by the wavelength λ, wavenumber vector $\vec{k}$ and the phase angle $\phi = 2\pi\,\frac{\Delta x}{\lambda}$. The schematic diagram shows the "wavefronts" of the periodic pattern.* (Figure based on B. Jähne "'Digitale Bildverarbeitung"' (S. 42).)

The periodic pattern is determined by the wavenumber vector $\vec{k} = [k_x, k_y]^T$ and the phase angle ϕ. The wavenumber vector $\vec{k}$ pointing in the direction of "Propagation direction of the wave" and has the length $|\vec{k}| = \frac{1}{\lambda}$.
As a consequence of this definition, the components k_x and k_y of the wavenumber vector $\vec{k}$ directly represent the number of wavelengths per unit length in that direction. An advantage of using the wavenumber vector is that it can be used in the same way to describe arbitrary dimensional periodic patterns.
For a complete description of the periodic pattern, in addition to wavenumber vector $\vec{k}$ and phase angle ϕ, the amplitude r is also required. Then the pattern or its gray value $f(\vec{x}) = f(x, y)$ at the position $\vec{x} = [x, y]^T$ can be described as follows:

$$f(\vec{x}) \;=\; r \cdot cos(2\pi\,\vec{k}^T\vec{x} - \phi) \tag{1}$$

Since the value range of the pattern of equation (ref eq: pattern) is in the interval $[-r, r]$, the equation for the practical application still has to be adapted to the interval $[0, 255]$ used:

$$f(x, y) \;=\; 127.5 + 127.5 \cdot cos(2\pi\,(k_x x + k_y y) - \phi) \tag{2}$$

Results of the application of this formula can be found in the figures 2 (a) and 3 (a).

Solution for task 4.1b

Task: *Implements a function that creates an image with a periodic pattern given by the wavenumber vector $\vec{k} = [k_x, k_y]^T$ and the phase angle ϕ!*

Definition of the interface:

```
// periodic pattern with wavenumber vector [kx, ky] and phase angle <phi>
// into the image <image>.
void wellenmuster (GrayImage& image, float kx, float ky, float phi);
```

Implementation:

```
//
// (4.1) Funktion zum Erzeugen eines periodischen Musters mit
//       Wellenzahlvektor [kx,ky] und Phasenverschiebung <phi>.
//
void wellenmuster (GrayImage& image, float kx, float ky, float phi)
{
    int     width  = image.getWidth();     // Breite des Bildes
    int     height = image.getHeight();    // Hoehe des Bildes
    float*  data   = image.getData();      // Zeiger auf Bilddaten

    // Grauwert fuer jede Position (x,y) entsprechend Gleichung (2) ermitteln
    for (int y=0; y<height; y++)
    {
        for (int x=0; x<width; x++)
        {
            data[y*width+x] = 127.5 + 127.5*cos(2*M_PI * (kx*x+ky*y) - phi);
        }
    }
}
```

Solution for task 4.2a

Task: *Which condition must be fulfilled when sampling an image signal?*

For correct sampling, the sampling theorem must be satisfied. This states that the sampling rate must be at least twice the maximum frequency present in the signal. In terms of sample step size r and wavelength λ this means:

$$r \leq \frac{1}{2}\lambda \tag{3}$$

Solution for task 4.2b

Task: *Writes a function that samples a picture at regular intervals r!*

Sampling is done by choosing each r th pixel of the input image as the sample point in both the x and y directions. Thus, the pixel (x, y) of the sampled image results from the pixel $(r \cdot x, r \cdot y)$ of the input image.

Definition of the interface:

```
// Sample the <input> image with step size <r>. Result in <output>.
void sample (GrayImage& input, GrayImage& output, int r);
```

Implementation:

```
//
// (4.2) Abtasten des <input> Bildes mit der Schrittweite <r>. Das
//       Ergebnis wird in das Bild <output> geschrieben.
//
void sample (GrayImage& input, GrayImage& output, int r)
{
    int    iwidth  = input.getWidth();     // Breites des Eingangsbildes
    int    iheight = input.getHeight();    // Hoehe des Eingangsbildes
    float* idata   = input.getData();      // Datenzeiger des Eingangsbildes

    int    owidth  = iwidth / r;           // Breites des Ausgangsbildes
    int    oheight = iheight / r;          // Hoehe des Ausgangsbildes
    float* odata   = output.getData();     // Datenzeiger des Ausgangsbildes

    // fuer jede Position im Ausgangsbild den entsprechenden Pixel des
    // Eingangsbildes bestimmen
    for (int y=0; y<oheight; y++)
    {
        for (int x=0; x<owidth; x++)
        {
            // Pixel (x,y) des Ausgangspixel entspricht Pixel (r*x,r*y) im Eingangsbild.
            // ACHTUNG: Die Indizes von Ein- und Ausgangsbild
            // muessen mit den jeweiligen Bildbreiten berechnet werden.
            odata[y*owidth+x] = idata[y*r * iwidth + x*r];
        }
    }
}
```

Solution for task 4.2c

Task: *Generates images with wavenumber vectors $\vec{k_1} = [0.21, 0.22]$ and $\vec{k_2} = [0.21, 0.24]$! Sample both images with the given distances!*

The distances are

$$\begin{aligned} r_1 &= 4 \\ r_2 &= 5 \end{aligned}$$

The results of the samples to be performed are shown in Figure 2. In order to clarify the qualitative change of the periodic pattern, the sampled images were stretched back to original size. There are significant qualitative changes in the form of altered wavelengths and directions of propagation of the pattern after sampling. These changes are not limited to a quality loss due to the lower resolution. The cause of the changes is the failure to comply with the sampling theorem.

$$\begin{aligned} \lambda_1 &= \frac{1}{\left|\binom{0.21}{0.22}\right|} &= 3.29 \ngeq 2 \cdot 4 \\ \lambda_2 &= \frac{1}{\left|\binom{0.21}{0.24}\right|} &= 3.14 \ngeq 2 \cdot 4 \end{aligned}$$

An example of the results of the sampling while maintaining the sampling theorem can be found in Figure. 3.

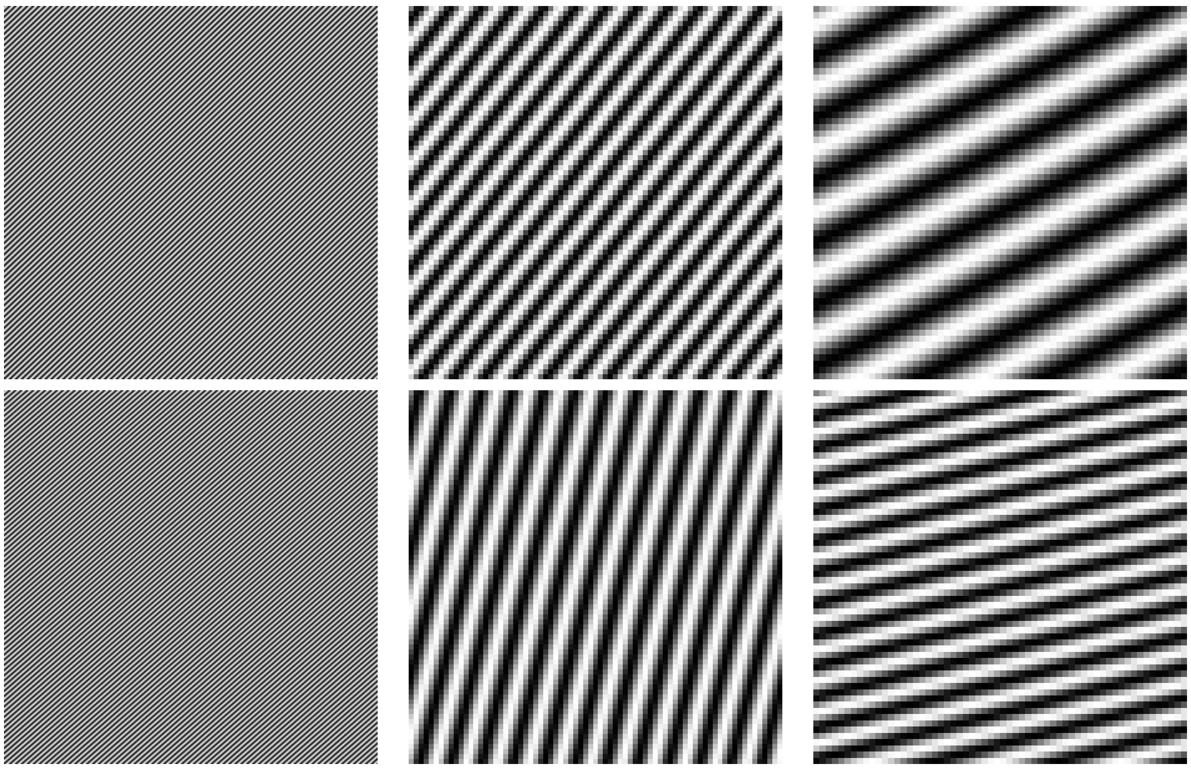
(a) (b) (c)

Figure 2: Periodic patterns with wavenumber vectors $\vec{k_1} = [0.21, 0.22]^T$ *(top)* and $\vec{k_2} = [0.21, 0.24]^T$ *(bottom)* , Shown is (a) the original periodic pattern, (b) the pattern is sampled with step size 4 and (c) the pattern is sampled with step size 5. Figures (b) and (c) were stretched to the size of the original for illustrationt.

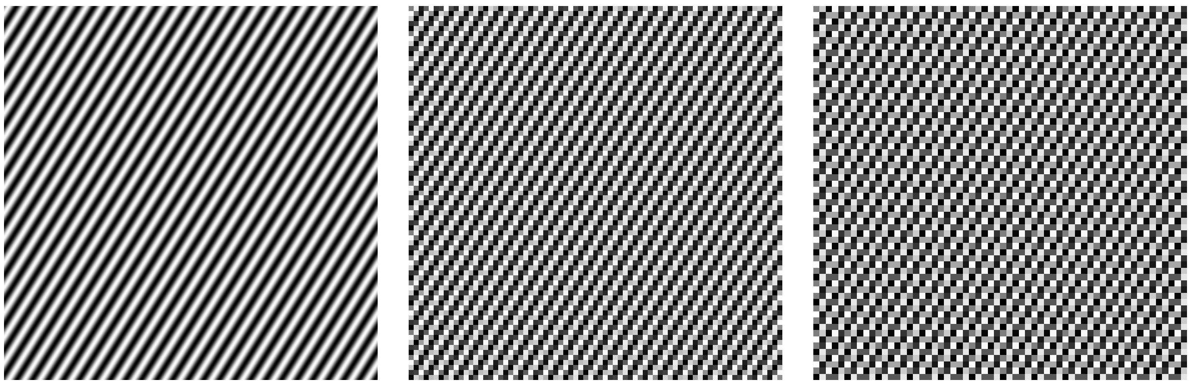
(a) (b) (c)

Figure 3: Sampling a periodic pattern with wavenumber vector $\vec{k} = [0.08, 0.05]$ with sampling theorem. (a) - (c) according to figure ref fig: results.

Solution for task 4.2d

Task: *What can be done to enforce the sampling theorem in natural images and avoid aliasing effects?*

For example, in order to force sampling of natural images at a given sampling rate to comply with the sampling theorem, the image may be filtered with a low-pass filter prior to sampling, thereby removing all frequencies that do not satisfy the sampling theorem.

Main program:

```
int main (int argc, char** argv)
{
    /* *********************************** */
    /* Uebung 04                           */
    /* *********************************** */

    /* *********************************** */
    /* Eingabe                             */
    /* *********************************** */

    int w,h;
    cout << "Geben Sie Breite und Höhe ein: " << endl;
    cin >> w >> h;

    float kx, ky;
    cout << "Geben Sie kx und ky des Wellenzahlvektors ein: " << endl;
    cin >> kx >> ky;

    int r;
    cout << "Geben Sie Abtastschrittweite ein: " << endl;
    cin >> r;

    /* *********************************** */
    /* Datenverarbeitung / Funktionsaufruf  */
    /* *********************************** */

    GrayImage image(w, h);
    GrayImage result(w/r, h/r);

        wellenmuster(image, kx, ky, 90);
    sample(image, result, r);

    /* *********************************** */
    /* Ausgabe                             */
    /* *********************************** */

    image.show();
    result.show(image.getWidth(), image.getHeight());

    /* *********************************** */
    /* Programmende                        */
    /* *********************************** */
    cout << "FINISHED.\n";
    return 1;
}
```