Example Solutions for Homework Assignment 6 (H6)

Problem 1 (Derivative Approximation)

(a) In order to determine the coefficients of the derivative mask, we have to perform a Taylor expansion for all neighbours of f_i that are relevant for our approximation filter. This yields

$$\begin{aligned}
f_{i-2} &= f_i - \frac{2}{1}hf_i' + \frac{4}{2}h^2 f_i'' - \frac{8}{6}h^3 f_i''' + \frac{16}{24}h^4 f_i^{(4)} - \frac{32}{120}h^5 f_i^{(5)} + \frac{64}{720}h^6 f_i^{(6)} + O(h^7) \\
f_{i-1} &= f_i - \frac{1}{1}hf_i' + \frac{1}{2}h^2 f_i'' - \frac{1}{6}h^3 f_i''' + \frac{1}{24}h^4 f_i^{(4)} - \frac{1}{120}h^5 f_i^{(5)} + \frac{1}{720}h^6 f_i^{(6)} + O(h^7) \\
f_i &= f_i \\
f_{i+1} &= f_i + \frac{1}{1}hf_i' + \frac{1}{2}h^2 f_i'' + \frac{1}{6}h^3 f_i''' + \frac{1}{24}h^4 f_i^{(4)} + \frac{1}{120}h^5 f_i^{(5)} + \frac{1}{720}h^6 f_i^{(6)} + O(h^7) \\
f_{i+2} &= f_i + \frac{2}{1}hf_i' + \frac{4}{2}h^2 f_i'' + \frac{8}{6}h^3 f_i''' + \frac{16}{24}h^4 f_i^{(4)} + \frac{32}{120}h^5 f_i^{(5)} + \frac{64}{720}h^6 f_i^{(6)} + O(h^7)
\end{aligned}$$

Since we are interested in computing an approximation to the first derivative, we have to choose the parameters $\alpha_{-2}, \ldots, \alpha_2$ in such a way that the following holds:

$$\begin{aligned}
0f_i + 0f_i' + 1f_i'' + 0f_i''' + 0f_i^{(4)} &\stackrel{!}{=} \alpha_{-2}f_{i-2} + \alpha_{-1}f_{i-1} + \alpha_0 f_i + \alpha_1 f_{i+1} + \alpha_2 f_{i+2} \\
&\approx (\alpha_{-2} + \alpha_{-1} + \alpha_0 + \alpha_1 + \alpha_2)f_i \\
&\quad +(-2\alpha_{-2} - \alpha_{-1} + \alpha_1 + 2\alpha_2)hf_i' \\
&\quad +(4\alpha_{-2} + \alpha_{-1} + \alpha_1 + 4\alpha_2)\frac{1}{2}h^2 f_i'' \\
&\quad +(-8\alpha_{-2} - \alpha_{-1} + \alpha_1 + 8\alpha_2)\frac{1}{6}h^3 f_i''' \\
&\quad +(16\alpha_{-2} + \alpha_{-1} + \alpha_1 + 16\alpha_2)\frac{1}{24}h^4 f_i^{(4)}\,.
\end{aligned}$$

This leads to the linear system

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ -2 & -1 & 0 & 1 & 2 \\ 4 & 1 & 0 & 1 & 4 \\ -8 & -1 & 0 & 1 & 8 \\ 16 & 1 & 0 & 1 & 16 \end{pmatrix} \begin{pmatrix} \alpha_{-2} \\ \alpha_{-1} \\ \alpha_0 \\ \alpha_1 \\ \alpha_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \frac{2}{h^2} \\ 0 \\ 0 \end{pmatrix} .$$

(b) Plugging the given coefficients into the Taylor expansion gives

$$\begin{aligned} & -\frac{1}{12h^2} f_{i-2} + \frac{16}{12h^2} f_{i-1} - \frac{30}{12} f_i + \frac{16}{12h^2} f_{i+1} - \frac{1}{12h^2} f_{i+2} \\ = \; & \underbrace{(-1+16-30+16-1)}_{=0} \frac{1}{12} \frac{1}{h^2} f_i \\ & + \underbrace{(2-16+16-2)}_{=0} \frac{1}{12} \frac{h}{h^2} f_i' \\ & + \underbrace{(-4+16+16-4) \frac{1}{12} \frac{1}{2} \frac{h^2}{h^2}}_{=1} f_i'' \\ & + \underbrace{(8-16+16-8)}_{=0} \frac{1}{12} \frac{1}{6} \frac{h^3}{h^2} f_i''' \\ & + \underbrace{(-16+16+16-16)}_{=0} \frac{1}{12} \frac{1}{24} \frac{h^4}{h^2} f_i^{(4)} \\ & + \underbrace{(32-16+16-32)}_{=0} \frac{1}{12} \frac{1}{120} \frac{h^5}{h^2} f_i^{(5)} \\ & + \underbrace{(-64+16+16-64)}_{=-96 \neq 0} \frac{1}{12} \frac{1}{720} \frac{h^6}{h^2} f_i^{(6)} \\ & + O(h^5) \\ = \; & f_i'' - \frac{h^4}{90} f_i^{(6)} + O(h^5) \\ = \; & f_i'' + O(h^4) \,, \end{aligned}$$

which shows that the order of consistency of the approximation is 4.

(c) If a derivative of order d is approximated with n points, we have (w.l.o.g.) n coefficients $\{\alpha_0, \dots, \alpha_{n-1}\}$ such that

$$\begin{aligned}&\alpha_0 f_i + \alpha_1 f_{i+1} + \cdots + \alpha_{n-1} f_{i+(n-1)} \\ &\quad = \quad \beta_0 f_i + \cdots + \beta_d h^d f_i^{(d)} + \cdots + \beta_{n-1} h^{n-1} f_i^{(n-1)} + \beta_n h^n f_i^{(n)} + \cdots,\end{aligned}$$

where it is assumed that $d \leq n-1$. The values β_k $(k = 0, 1, \dots)$ are linear combinations of the coefficients α_i $(i = 0, \dots, n-1)$ (See previous a) and b)). Since we are approximating the $d{-}th$ derivative of f_i, we know that $\forall i$, $\alpha_i \sim \frac{1}{h^d}$. Thus, it must hold that $\beta_d = \frac{1}{h^d}$, and $\beta_k = \frac{0}{h^d}$, $\forall k \in \{0, \dots, n-1\} \setminus \{d\}$. The approximation error depends on the values β_k for $k \geq n$. In particular, the lower bound of the order of consistency is obtained when $\beta_n \sim \frac{1}{h^d} \neq 0$. Without loss of generality, let $\beta_k := \frac{\tilde{\beta}_k}{h^d}$ for $k \geq n$. In this case, the approximation reads

$$\begin{aligned}&\alpha_0 f_i + \alpha_1 f_{i+1} + \cdots + \alpha_{n-1} f_{i+(n-1)} \\ &\quad = \quad f_i^{(d)} + \tilde{\beta}_n \frac{h^n}{h^d} f_i^{(n)} + \tilde{\beta}_{n+1} \frac{h^{n+1}}{h^d} f_i^{(n+1)} + \dots \\ &\quad = \quad f_i^{(d)} + O(h^{n-d}),\end{aligned}$$

which shows that the order of consistency is at least $n - d$.

Problem 2 (Fourier Analysis of Derivative Filters)

(a) We compute the Fourier transform of the function

$$g(x) = \frac{f(x-2h) - f(x-h) - f(x+h) + f(x+2h)}{3h^2}$$

using the shift theorem:

$$\begin{aligned}\mathcal{F}[g](u) &= \frac{1}{3h^2} \left(\mathcal{F}[f(x-2h)](u) - \mathcal{F}[f(x-h)](u) - \mathcal{F}[f(x+h)] + \mathcal{F}[f(x+2h)]\right) \\ &= \frac{1}{3h^2} \left(e^{4\pi i h u} - e^{2\pi i h u} - e^{-2\pi i h u} + e^{-4\pi i h u}) \mathcal{F}[f(x)](u)\right) \\ &= \left(\frac{2\cos(4\pi h u)}{3h^2} - \frac{2\cos(2\pi h u)}{3h^2}\right) \mathcal{F}[f](u)\end{aligned}$$

(b) The computation of a Taylor expansion around 0 gives us

$$\begin{aligned}
\cos(2\pi hu) &= \frac{\cos(0)}{0!} - \frac{\sin(0)}{1!}2\pi hu - \frac{\cos(0)}{2!}(2\pi hu)^2 \\
&\quad + \frac{\sin(0)}{3!}(2\pi hu)^3 + \frac{\cos(0)}{4!}(2\pi hu)^4 + \mathcal{O}(h^5u^5) \\
&= 1 - 2\pi^2h^2u^2 + \frac{2}{3}\pi^4h^4u^4 + \mathcal{O}(h^5u^5) \\
\cos(4\pi hu) &= \frac{\cos(0)}{0!} - \frac{\sin(0)}{1!}4\pi hu - \frac{\cos(0)}{2!}(4\pi hu)^2 \\
&\quad + \frac{\sin(0)}{3!}(4\pi hu)^3 + \frac{\cos(0)}{4!}(4\pi hu)^4 + \mathcal{O}(h^5u^5) \\
&= 1 - 8\pi^2h^2u^2 + \frac{32}{3}\pi^4h^4u^4 + \mathcal{O}(h^5u^5)
\end{aligned}$$

Let us now use this Taylor expansion to compute the order of approximation explicitly. The derivation rule for the Fourier transform yields:

$$\mathcal{F}[f''](u) = (i2\pi u)^3\mathcal{F}[f](u) = -4\pi^2u^2\mathcal{F}[f](u)$$

and thus, using additionally the Taylor-expansion from (b), we get:

$$\begin{aligned}
\mathcal{F}[g](u) - \mathcal{F}[f''](u) &= \frac{2}{3h^2}(\cos(4\pi hu) - \cos(2\pi hu))\mathcal{F}[f](u) + 4\pi^2u^2\mathcal{F}[f](u) \\
&= \left(\frac{1}{h^2}(-4\pi^2h^2u^2 + \frac{20}{3}\pi^4h^4u^4 + \mathcal{O}(h^5u^5)) + 4\pi^2u^2\right)\mathcal{F}[f](u) \\
&= \frac{20}{3}\pi^4h^2u^4\mathcal{F}[f](u) + \mathcal{O}(h^3u^5)
\end{aligned}$$

Thus, we get a consistency order of 2, which is the same as we would obtain in the spatial approximation analysis. However, the dependency of the error term on the frequency u yields additional information.

(c) The approximation error will increase for higher frequencies, since the error terms in $\mathcal{O}(h^2u^2)$ depend clearly on u and will thus become larger for higher frequencies.

Problem 3 (Linear Filters)

(a) In the lecture, the highpass and bandpass filter are defined as substractions of lowpass filters and the identity. If two LSI filters A and B

are given, one can, due to the superposition principle, also express the result of the filter $A - B$ as the difference of filtered images:

$$(A - B)u = Au - Bu.$$

This is what we exploit for a simple implementation of highpass and bandpass filters. The code for the routine `highpass` is given by

```
/*--------------------------------------------------------------------*/

void highpass(float sigma, /* standard deviation of Gaussian */
              long nx,     /* image dimension in x direction */
              long ny,     /* image dimension in y direction */
              float **f)   /* input: original image ; output: highpass filtered*/
/* highpass filter */
{
  int i,j;       /* loop variables */
  float **gauss; /* Gaussian smoothed image */

  /* allocate memory */
  alloc_matrix(&gauss,nx+2,ny+2);

  /* copy original image to temporary array */
  for (i=1;i<=nx;i++)
    for (j=1;j<=ny;j++)
      gauss[i][j]=f[i][j];

  /* apply Gaussian to temporary array */
  gauss_conv(sigma,nx,ny,GRID_SIZE,GRID_SIZE,PRECISION,BOUNDARY,gauss);

  /* compute highpass filter */
  for (i=1;i<=nx;i++)
    for (j=1;j<=ny;j++) {
      /* Supplement your own code here */
      f[i][j] -= gauss[i][j];
    }

  /* free memory */
  disalloc_matrix(gauss,nx+2,ny+2);
}

/*--------------------------------------------------------------------*/
```

For the routine `bandpass` the following code is obtained:

```
/*--------------------------------------------------------------------*/

void bandpass(float sigma1, /* standard deviation of first Gaussian */
              float sigma2, /* standard deviation of second Gaussian */
                            /* sigma1 > sigma2 */
              long nx,     /* image dimension in x direction */
              long ny,     /* image dimension in y direction */
              float **f)   /* input: original image ; output: bandpass
                              filtered */
/* bandpass filter */
{
  int i,j;        /* loop variables */
  float **gauss1; /* Gaussian smoothed image (stddev sigma1) */
```

```
  float **gauss2; /* Gaussian smoothed image (stddev sigma2) */

  /* allocate memory */
  alloc_matrix(&gauss1,nx+2,ny+2);
  alloc_matrix(&gauss2,nx+2,ny+2);

  /* copy original image to temporary arrays */
  for (i=1;i<=nx;i++)
    for (j=1;j<=ny;j++) {
      gauss1[i][j]=f[i][j];
      gauss2[i][j]=f[i][j];
    }

  /* apply Gaussian to temporary arrays */
  gauss_conv(sigma1,nx,ny,GRID_SIZE,GRID_SIZE,PRECISION,BOUNDARY,gauss1);
  gauss_conv(sigma2,nx,ny,GRID_SIZE,GRID_SIZE,PRECISION,BOUNDARY,gauss2);

  /* compute bandpass filter */
  for (i=1;i<=nx;i++)
    for (j=1;j<=ny;j++) {
      /* Supplement your own code here */
      f[i][j]=gauss2[i][j]-gauss1[i][j];
    }

  /* free memory */
  disalloc_matrix(gauss1,nx+2,ny+2);
  disalloc_matrix(gauss2,nx+2,ny+2);
}

/*--------------------------------------------------------------------*/
```

Note that the order of Gaussians here makes a difference and that the basic idea of substracting the narrow kernel from the broad kernel references the sizes in the Fourier domain.

(b)

- In order to remove the high-frequent noise in `pruebab1.pgm`, we just have to apply a lowpass filter, i.e. Gaussian convolution, to taste. A possible parameter choice is $\sigma = 5$ (see Figure 1).
- Background structures are lowfrequent and thus a highpass filter should be applied to `angiogram.pgm`. We choose here $\sigma = 2$ and obtain the result in Figure 2.
- Finally, we can isolate the dark gaps between the tiles in `tile.pgm` by designing a suitable bandpass filter (e.g. $\sigma_1 = 8, \sigma_2 = 2$). Since the gaps are mapped close to zero, it's best to compare the inverse of the original and the filtered image. In particular, due to the linearity of the filter, the mapping of gaps to bright structures and tile pixels to values close to zero can be obtained by swapping the roles of σ_1 and σ_2.

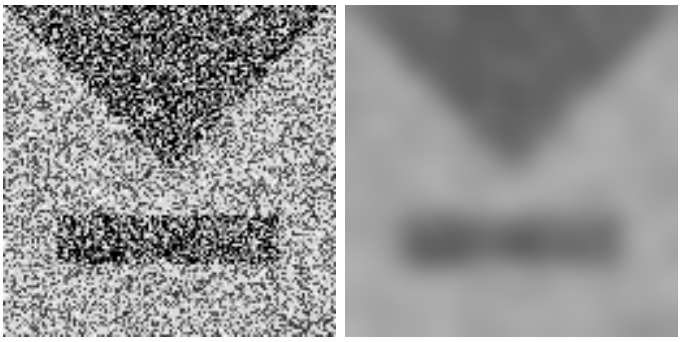

Figure 1: *Left:* `pruebab1.pgm`: *Right:* lowpass filter result with $\sigma = 5$.

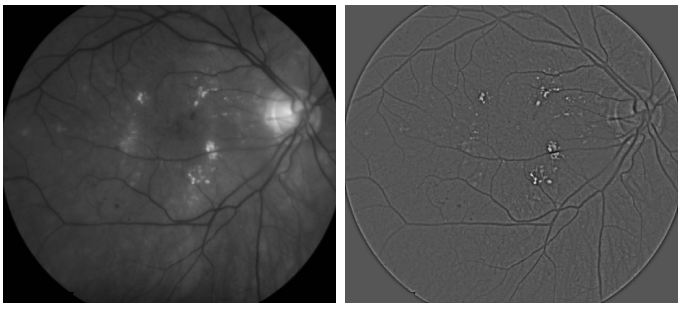

Figure 2: *Left:* `angiogram.pgm`: *Right:* highpass filter result with $\sigma = 2$.

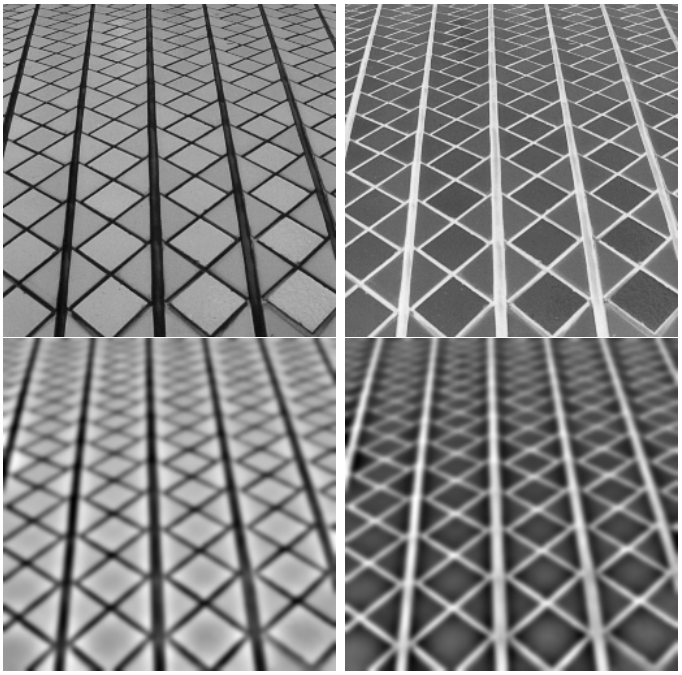

Figure 3: *Top:* `tile.pgm` and its inverse *Right:* bandpass filter result with $\sigma_1 = 8$, $\sigma_2 = 2$ and its inverse.