Image Processing and Computer Vision 2007/08
Example Solutions for Theoretical Assignments 3 (T3)

Problem 1 (Interpolation)

(a) The synthesis function β_5 at integer values gives

$$\beta_5(k) = \begin{cases} \frac{66}{120}, & k = 0 \\ \frac{26}{120}, & k = \pm 1 \\ \frac{1}{120}, & k = \pm 2 \\ 0, & \text{else}\,. \end{cases}$$

The coefficients $c_1, \ldots, c_N$ are determined solving the linear system of equations:

$$\begin{pmatrix} \frac{66}{120} & \frac{26}{120} & \frac{1}{120} & 0 & \cdots & \cdots & \cdots & 0 \\ \frac{26}{120} & \frac{66}{120} & \frac{26}{120} & \frac{1}{120} & 0 & \cdots & \cdots & 0 \\ \frac{1}{120} & \frac{26}{120} & \frac{66}{120} & \frac{26}{120} & \frac{1}{120} & 0 & \cdots & 0 \\ & \ddots & \ddots & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \ddots & \ddots & \\ 0 & \cdots & 0 & \frac{1}{120} & \frac{26}{120} & \frac{66}{120} & \frac{26}{120} & \frac{1}{120} \\ 0 & \cdots & \cdots & 0 & \frac{1}{120} & \frac{26}{120} & \frac{66}{120} & \frac{26}{120} \\ 0 & \cdots & \cdots & \cdots & 0 & \frac{1}{120} & \frac{26}{120} & \frac{66}{120} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ \vdots \\ c_{N-2} \\ c_{N-1} \\ c_N \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ \vdots \\ f_{N-2} \\ f_{N-1} \\ f_N \end{pmatrix}$$

(b) We first compute the interpolation coefficients c_1, c_2, c_3 for *cubic B-spline interpolation* solving the linear system of equations

$$\begin{pmatrix} \frac{2}{3} & \frac{1}{6} & 0 \\ \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \\ 0 & \frac{1}{6} & \frac{2}{3} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix} = \begin{pmatrix} 4 \\ 28 \\ 24 \end{pmatrix},$$

obtaining $c_1 = -3$, $c_2 = 36$, $c_3 = 27$. Then, the function $f(x)$ can be interpolated via the formula

$$f(x) = \sum_{k \in \{1,2,3\}} c_k \, \beta_3(x - k)\,.$$

Using the symmetry condition $\beta_3(z) = \beta_3(-z)$, at the points $x = 1/2$, and $x = 3/2$ we have

$$\begin{aligned} f(1/2) &= c_1\,\beta_3(-1/2) + c_2\,\beta_3(-3/2) + c_3\,\beta_3(-5/2) \\ &= c_1\,\beta_3(1/2) + c_2\,\beta_3(3/2) + c_3\,\beta_3(5/2) \\ &= -\tfrac{33}{48} \\ &= -0.6875\,, \\ f(3/2) &= c_1\,\beta_3(1/2) + c_2\,\beta_3(-1/2) + c_3\,\beta_3(-3/2) \\ &= c_1\,\beta_3(1/2) + c_2\,\beta_3(1/2) + c_3\,\beta_3(3/2) \\ &= \tfrac{786}{48} \\ &= 16.375\,, \end{aligned}$$

with

$$\begin{aligned} \beta_3(1/2) &= \tfrac{2}{3} - \left(\tfrac{1}{2}\right)^2 + \tfrac{1}{2}\,\left(\tfrac{1}{2}\right)^3 = \tfrac{23}{48}\,, \\ \beta_3(3/2) &= \tfrac{1}{6}\,\left(2 - \tfrac{3}{2}\right)^3 = \tfrac{1}{48}\,. \end{aligned}$$

Here, one should note that determining the function value at the point $\beta_3(1/2)$ is actually extrapolation. In general, extrapolation with B-splines not very useful. This is due to the fact that B-splines have only a small support, and outside of the interval in which the function values are given, the number of B-splines contributing to the value of a point is decreasing. Thus, the function values of extrapolated points tend to zero and finally become zero if they are too far away from this interval.

Problem 2 (Multiple Choice)

1. **The discrete Wavelet transform has only optimal (linear) complexity in 1-D.**

   Truse/False. If the word "only" is understood with respect to the degree of complexity, the statement is true. This can be easily verified as follows: If applying the 1-D Fast Wavelet transform to a 1-D signal of size $n = 2^k$ one has to compute in total

$$\begin{aligned}
\sum_{l=0}^{k-1} \left(\frac{1}{2}\right)^l \cdot n &= \left(\sum_{l=0}^{k-1} \left(\frac{1}{2}\right)^l\right) \cdot n \\
&= \left(\sum_{l=0}^{\infty} \left(\frac{1}{2}\right)^l - \sum_{l=k}^{\infty} \left(\frac{1}{2}\right)^l\right) \cdot n \\
&= \left(\sum_{l=0}^{\infty} \left(\frac{1}{2}\right)^l - \frac{1}{2^k} \sum_{l=0}^{\infty} \left(\frac{1}{2}\right)^l\right) \cdot n \\
&= (2 - \frac{1}{n} \cdot 2) \cdot n \\
&= 2 \cdot n - 2 \\
&= 2\,(n-1)
\end{aligned}$$

   coefficients. Moreover, one needs two operations per coefficient per level – one addition or subtraction and one division for scaling. This yields in total $4\,(n-1)$ operations for all levels which gives linear complexity. However, if the word "only" is understood with respect to the dimension, the statement says that the wavelet transform is not linear any more in higher dimensions. This statement is evidently false.Analogously to the 1-D case this can also be shown for 2-D images: If applying the 2-D Fast Wavelet transform to a 2-D image of

size $n = 2^k \times 2^k$ one has to compute in total

$$\begin{aligned}
\sum_{l=0}^{k-1} \left(\frac{1}{2} \cdot \frac{1}{2}\right)^l \cdot n &= \left(\sum_{l=0}^{k-1} \left(\frac{1}{2} \cdot \frac{1}{2}\right)^l\right) \cdot n \\
&= \left(\sum_{l=0}^{\infty} \left(\frac{1}{2} \cdot \frac{1}{2}\right)^l - \sum_{l=k}^{\infty} \left(\frac{1}{2} \cdot \frac{1}{2}\right)^l\right) \cdot n \\
&= \left(\sum_{l=0}^{\infty} \left(\frac{1}{2} \cdot \frac{1}{2}\right)^l - \left(\frac{1}{2^k} \cdot \frac{1}{2^k}\right) \sum_{l=0}^{\infty} \left(\frac{1}{2} \cdot \frac{1}{2}\right)^l\right) \cdot n \\
&= (\frac{4}{3} - \frac{1}{n} \cdot \frac{4}{3}) \cdot n \\
&= \frac{4}{3} \cdot n - \frac{4}{3} \\
&= \frac{4}{3}(n-1)
\end{aligned}$$

coefficients. Please note that $\frac{4}{3}(n-1) \in \mathbb{N}$ for $n = 2^k \cdot 2^k = 4^k$ with $k \in \mathbb{N}$. Due to the **separability** we only need four operations per coefficient per level – one addition or subtraction and one division for scaling and this for two directions. This yields in total $\frac{16}{3}(n-1)$ operations which still gives **linear** complexity!

2. **For the German word "Lagerregal" the Huffman coding does not allow a better compression that any other coding scheme that assigns the codes 00, 01, 10, 110 and 111 arbitrarily to the letters.**

   True. As one can easily verfiy, all letters in the German word "Lagerregal" occur twice. Because of this fact, any mapping of codes to letters will result in the same compression rate. Note that the achieved compression rates will be equal, but not the code words! See figure 1 for an example.

Letter	Code
L	000
A	001
G	01
E	10
R	11

Table 1: Mappings of letters to codes

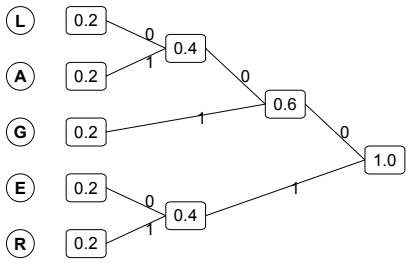

Figure 1: Huffman tree example.

3. **The compression ratio for an alternating signal $f = (7, 8, \ldots, 7, 8)$ can be improved signifcantly, if one uses bitplane coding instead of the ordinary run length encoding.**

   True/False. First, note that $\langle 7 \rangle_{10} = \langle 111 \rangle_2$ and $\langle 8 \rangle_{10} = \langle 1000 \rangle_2$. So, between each two entries the last 4 bits flip. If we assume bytewise coding, the four most significant bits of our signal do not change over the whole signal. Now, while RLE cannot profit from that fact, bitplane coding can exploit this constellation extensively, because the top 4 bits of the whole signal being all equal to zero can be merged plane by plane using RLE. Furthermore, regarding the mentioned Gray code, one should see that this could lead again to higher compression rates. In Gray code, between two consecutive numbers only one bit changes. So 7 bits could be treated efficiently (using RLE). If one assumes, however, that only four bit are stored per pixel, we have bit changes in all four channels. In this case also the bitplane coding does not give any advantage.

4. **Quadratic B-splines require less computational effort for determining the coeffcients than cubic B-splines.**

   False. When determining the coefficients, what has to be done is to make sure that the interpolated (reconstructed) signal conincides with the discrete samples at all positions where discrete samples were given, i.e.

$$\sum_{k=1}^{N} c_k \; \varphi(i-k) = f_i \qquad \text{for } i = 1, \ldots, N$$

Now, regarding the values of β_2 and β_3 at integer positions, we have

$$\beta_2(k) = \begin{cases} \frac{3}{4} & , k = 0 \\ \frac{1}{8} & , k \in \{-1, 1\} \\ 0 & , k \in \mathbb{Z} \setminus \{-1, 0, 1\} \end{cases}$$

$$\beta_3(k) = \begin{cases} \frac{2}{3} & , k = 0 \\ \frac{1}{6} & , k \in \{-1, 1\} \\ 0 & , k \in \mathbb{Z} \setminus \{-1, 0, 1\} \end{cases}$$

One can see that both, quadratic and cubic splines only contribute at integer positions -1,0 and 1. Hence the resulting linear systems of equations are both tridiagonal and can thus be both solved in linear time with the Thomas algorithm.

5. **Histogram equalisation is a point operation that is invertible by construction.**

   False. As an easy counterexample, consider the signal used in the classroom assignment. After the values 1,2 and 3 have been merged onto one stack, the inverse direction is obviously nonunique.
   Mathematically speaking, for a given histogram $f \in \mathbb{R}^n$ we can specify a matrix $H \in \mathbb{R}^{n \times n}$ that performs the histogram equalisation. If histogram equalisations were invertible, then H would have to be invertible as well. Consider the example from the classroom assignment,

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

   For the histogram of the Classroom work $f = (3, 1, 1, 4, 11, 9, 13, 8)$ computing Hf leads to the equalised histogram $Hf = (5, 4, 0, 11, 9, 0, 13, 8)$. Since for $x = (0, 0, 1, 0, 0, 0, 0, 0)$ we have that $Hx = 0x$. As a consequence, 0 is an eigenvalue of H and thus H is not invertible.

6. **The subsequent application of a lowpass and highpass filter always yields the original signal.**

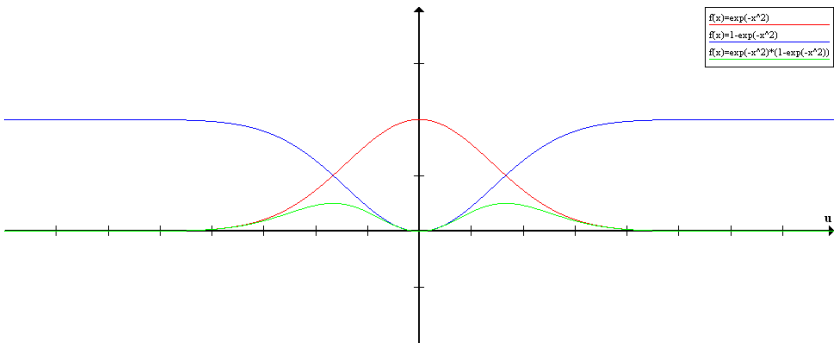

Figure 2: Fourier spectra of filters. **Red line:** Gaussian shaped lowpass filter. **Blue line:** An example for an highpass filter. **Green line:** Product of given lowpass and highpass filter. The result is a bandpass filter.

False. To explain this, consider a *lowpass* filter, for example convolution with a Gaussian kernel. The important detail here is that while it allows low frequencies to pass, high frequencies are eliminated. One could speak of a *high-non-pass* filter to stress this property. As a consequence, a lowpass filter decreases the information content of an image. The same holds for a highpass (or *low-non-pass*) filter. It eliminates information as well. The subsequent application of highpass and lowpass filters hence deletes information with each application. The result of such a filter is also called bandpass filter, a filter that only lets a certain frequency band survive. Consider figure 2 as an example. In Fourier domain, spatial convolutions are carried out as simple multiplications. As a consequence, the effect of subsequent convolutions (once with a highpass and once with a lowpass filter) can be characterized by multiplying the corresponding kernels.

Problem 3 (Linear Filters)

(a) In order to determine the type of our filter, we decompose the given stencil as follows:

$$\frac{1}{12}\begin{pmatrix}1 & 2 & 1\\ 1 & 2 & 1\\ 1 & 2 & 1\end{pmatrix} = \underbrace{\frac{1}{3}\begin{pmatrix}0 & 1 & 0\\ 0 & 1 & 0\\ 0 & 1 & 0\end{pmatrix}}_{\substack{\text{lowpass}\\ \text{in } y\text{-direction}}} \underbrace{\frac{1}{4}\begin{pmatrix}0 & 0 & 0\\ 1 & 2 & 1\\ 0 & 0 & 0\end{pmatrix}}_{\substack{\text{lowpass}\\ \text{in } x\text{-direction}}}$$

- x-direction: Binomial filter $\frac{1}{4}$ (1 2 1) $\Rightarrow$ **lowpass filter**
- y-direction: Box filter $\frac{1}{3}$ $(1\ 1\ 1)^\top$ $\Rightarrow$ **lowpass filter**

(b) Again we have to distinguish between x- and y-direction. We write our filter as follows:

$$\begin{aligned}\tfrac{1}{12}\begin{pmatrix}-1 & -2 & -1\\ 2 & 4 & 2\\ -1 & -2 & -1\end{pmatrix} &= \tfrac{1}{3}\begin{pmatrix}0 & -1 & 0\\ 0 & 2 & 0\\ 0 & -1 & 0\end{pmatrix} \tfrac{1}{4}\begin{pmatrix}0 & 0 & 0\\ 1 & 2 & 1\\ 0 & 0 & 0\end{pmatrix}\\ &= \underbrace{\left(\underbrace{\begin{pmatrix}0 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 0\end{pmatrix}}_{\text{identity}} - \underbrace{\frac{1}{3}\begin{pmatrix}0 & 1 & 0\\ 0 & 1 & 0\\ 0 & 1 & 0\end{pmatrix}}_{\substack{\text{lowpass}\\ \text{in } y\text{-direction}}}\right)}_{\substack{\text{highpass}\\ \text{in } y\text{-direction}}} \underbrace{\frac{1}{4}\begin{pmatrix}0 & 0 & 0\\ 1 & 2 & 1\\ 0 & 0 & 0\end{pmatrix}}_{\substack{\text{lowpass}\\ \text{in } x\text{-direction}}}\end{aligned}$$

- x-direction: Binomial filter $\frac{1}{4}$ (1 2 1) $\Rightarrow$ **lowpass filter**
- y-direction: Identity - lowpass filter $\Rightarrow$ **highpass filter**

(c) The last stencil can be decomposed into:

$$\frac{1}{256}\begin{pmatrix} -1 & -4 & -6 & -4 & -1 \\ -4 & 0 & 8 & 0 & -4 \\ -6 & 8 & 28 & 8 & -6 \\ -4 & 0 & 8 & 0 & -4 \\ -1 & -4 & -6 & -4 & -1 \end{pmatrix}$$

$$= \underbrace{\underbrace{\frac{1}{16}\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 2 & 4 & 2 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}}_{\text{2-D lowpass}} - \underbrace{\frac{1}{256}\begin{pmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{pmatrix}}_{\text{2-D lowpass}}}_{\text{2-D bandpass}}$$

- x- and y-direction: We see that the filter can be written as difference of two 2-D lowpass filters $\Rightarrow$ **2-D bandpass filter**.