Image Processing and Computer Vision (IPCV)

Prof. J. Weickert
Mathematical Image Analysis Group

Summer term 2019
Saarland University

Example Solutions for Homework Assignment 8 (H8)

Problem 1 (Multiple Choice)

1. *By multiplying the Fourier transform of an image iteratively with a sinc function, one obtains a Gaussian smoothed image in the spatial domain.*

   True. Multiplying in the Fourier domain by a sinc function corresponds to convolving in the spatial domain with a box function. Since iterative convolutions with a box function in the spatial domain will approximate a convolution with a Gaussian, we can apply Gaussian smoothing this way.

2. *Huffman coding is most efficient if a word consists of letters that all occur with the same frequency.*

   False. The basic principle of Huffman coding is, that letters with high occurrence frequency are represented by a shorter code than letters that appear less frequently. Thus, if all letters occur with the same or very similar frequency, Huffman coding will not be very effective.

3. *Discrete histogram equalisation is always invertible.*

   False. In general, discrete histogram equalisation is not invertible, since the mapping of several grey values to a single new one leads to a loss of information.

4. *If the discrete cosine transform of a signal f is nonsymmetric, then f has a nonzero imaginary part.*

   False. There is no relation between the symmetry of $DCT(f)$ and the imaginary part of f. We can see this in two ways:

   (a) We defined the DCT as a mapping from the real numbers to the real numbers. No matter what real function $DCT(f)$ is, f is also real.

(b) We can consider a counterexample. In lecture 6, on slide 6-7 we compute the DCT of the signal $\boldsymbol{f} = (6, 4, 5, 1)^T$ and get

$$DCT(\boldsymbol{f}) = (8, 2.994, -1, 2.008)^T.$$

Here we have a real signal whose DCT is nonsymmetric.

5. *Taking the forward differences of the backward differences of a 1D signal approximates its second derivative.*

   True. A simple computation yields

$$\begin{aligned} u &= \ldots, u_{i-1}, u_i, u_{i+1}, \ldots \\ BD(u) &= \ldots, \frac{u_{i-1} - u_{i-2}}{h}, \frac{u_i - u_{i-1}}{h}, \frac{u_{i+1} - u_i}{h}, \ldots \\ FD(BD(u)) &= \ldots, \frac{u_i - 2u_{i-1} + u_{i-2}}{h^2}, \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2}, \frac{u_{i+2} - 2u_{i+1} + u_i}{h^2}, \ldots \end{aligned}$$

   which is a well known approximation scheme for the second order derivative.

6. *Hard wavelet shrinkage without cycle spinning is idempotent.*

   True. In hard wavelet shrinkage, every coefficient below a threshold T are set to zero. Iterating this process does not change anything after the first iteration, because all coefficients below T have already been set to zero.

Problem 2 (Nonlinear Diffusion)

(a) To get the stencil we reformulate the given formula of the explicit scheme by factoring out the pixel values of the old time step k:

$$\begin{aligned}
u_i^{k+1} &= u_i^k + \frac{\tau}{h}\left(g_{i+1/2}^k \frac{u_{i+1}^k - u_i^k}{h} - g_{i-1/2}^k \frac{u_i^k - u_{i-1}^k}{h}\right) \\
&= u_{i-1}^k \underbrace{\left(\frac{\tau}{h^2} g_{i-1/2}^k\right)}_{=:l_i} \quad + \quad u_i^k \cdot \underbrace{\left(1 - \frac{\tau}{h^2} g_{i+1/2}^k - \frac{\tau}{h^2} g_{i-1/2}^k\right)}_{=:c_i} \\
&\quad + \quad u_{i+1}^k \underbrace{\left(\frac{\tau}{h^2} g_{i+1/2}^k\right)}_{=:r_i}
\end{aligned}$$

Thus the stencil is

$$\boxed{l_i \mid c_i \mid r_i} = \boxed{\frac{\tau}{h^2} g_{i-1/2}^k \;\Big|\; 1 - \frac{\tau}{h^2} g_{i+1/2}^k - \frac{\tau}{h^2} g_{i-1/2}^k \;\Big|\; \frac{\tau}{h^2} g_{i+1/2}^k}$$

(b) Considering the stencil we get the following system of equations for one iteration step:

$$\begin{aligned}
u_1^{k+1} &= l_1 u_0^k + c_1 u_1^k + r_1 u_2^k \overset{u_0^k = u_1^k}{=} (l_1 + c_1) u_1^k + r u_2^k \\
u_2^{k+1} &= l_2 u_1^k + c_2 u_2^k + r_2 u_3^k \\
u_3^{k+1} &= l_3 u_2^k + c_3 u_3^k + r_3 u_4^k \\
&\vdots \\
u_{N-1}^{k+1} &= l_{N-1} u_{N-2}^k + c_{N-1} u_{N-1}^k + r_{N-1} u_N^k \\
u_N^{k+1} &= l_N u_{N-1}^k + c_N u_N^k + r_N u_{N+1}^k \overset{u_0^k = u_1^k}{=} l_N u_{N-1}^k + (c_N + r_N) u_N^k
\end{aligned}$$

We want to formulate this system as a single matrix-vector product:

$$\underbrace{\begin{pmatrix} u_1^{k+1} \\ \vdots \\ u_N^{k+1} \end{pmatrix}}_{\boldsymbol{u}^{k+1}} = \underbrace{\begin{pmatrix} q_{1,1} & \cdots & q_{1,N} \\ \vdots & \ddots & \vdots \\ q_{N,1} & \cdots & q_{N,N} \end{pmatrix}}_{Q(\boldsymbol{u}^k)} \underbrace{\begin{pmatrix} u_1^k \\ \vdots \\ u_N^k \end{pmatrix}}_{\boldsymbol{u}^k}$$

For that purpose we define $Q(\boldsymbol{u}^k)$ to be

$$Q(\boldsymbol{u}^k) := \begin{pmatrix} l_1 + c_1 & r_1 & & & & \\ l_2 & c_2 & r_2 & & & \\ & l_3 & c_3 & r_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & l_{N-1} & c_{N-1} & r_{N-1} \\ & & & & l_N & c_N + r_N \end{pmatrix}$$

where non-given entries are 0.

$Q(\boldsymbol{u}^k)$ is a symmetric, tridiagonal matrix, as $l_{i+1} = \tau g^k_{(i+1)-1/2} = \tau g^k_{i+1/2} = r_i$. Considering its entries we clearly see how it is related to the stencil of problem (a).

(c) Now we check for unit row and unit column sums.

- **Unit row sums:**
  $\forall i \in \{1, \ldots, N\}$:

$$\begin{aligned} \sum_{j=1}^{N} q_{i,j} &= l_i + c_i + r_i \\ &= (\frac{\tau}{h^2} g^k_{i-1/2}) + (1 - \frac{\tau}{h^2} g^k_{i+1/2} - \frac{\tau}{h^2} g^k_{i-1/2}) + (\frac{\tau}{h^2} g^k_{i+1/2}) \\ &= 1 \end{aligned}$$

- **Unit column sums:**
  As $Q(\boldsymbol{u}^k)$ is symmetric, column sums are equal to the row sums:
  $\forall j \in \{1, \ldots, N\}$:

$$\sum_{i=1}^{N} q_{i,j} = \sum_{i=1}^{N} q_{j,i} = 1$$

So we can state that nonlinear diffusion preserves the average grey level. However, the maximum-minimum principle is not necessarily satisfied as the diagonal entries $q_{i,i}$ $(i = 1 \ldots N)$ might be negative. One could impose that $c_i \geq 0$. That means

$$\begin{aligned}
& c_i \geq 0 \\
\Leftrightarrow \quad & 1 - \frac{\tau}{h^2} g^k_{i+1/2} - \frac{\tau}{h^2} g^k_{i-1/2} \geq 0 \\
\Leftrightarrow \quad & \tau \leq \frac{h^2}{g^k_{i+1/2} + g^k_{i-1/2}} \\
\overset{|g_i| \leq 1}{\Rightarrow} \quad & \tau \leq \frac{h^2}{1+1} = \frac{1}{2} h^2
\end{aligned}$$

Thus the maximum-minimum principle is satisfied if $\tau \leq \frac{1}{2}h^2$ and $|g_i| \leq 1$.

Problem 3 (Wavelet Shrinkage)

(a) Hard shrinkage is a simple thresholding of the wavelet coefficients, while soft and Garrote shrinkage also rescale the coefficients that are not set to 0. For all three cases one should avoid to modify the scaling coefficient with array indices (0,0).

```
/*--------------------------------------------------------------------------*/
void hard_shrinkage(int nx, int ny, float** coefficients, float threshold) {
  int i,j;
  int count = 0;

  for(j=0; j<nx; j++)
    for(i=0; i<ny; i++) {
      /* Supplement your own code here */
      if (fabsf (coefficients[i][j]) <= threshold &&
          (i>0 || j>0)) {
        coefficients[i][j] = 0.0;
        count++;
      }
    }

  return;
}

/*--------------------------------------------------------------------------*/
void soft_shrinkage(int nx, int ny, float** coefficients, float threshold) {
  int i,j;
  int count = 0;

  for(j=0; j<nx; j++)
    for(i=0; i<ny; i++) {
      /* Supplement your own code here */
      if (i>0 || j>0) {
        if (fabsf (coefficients[i][j]) <= threshold) {
          coefficients[i][j] = 0.0;
          count++;
        } else {
          if (coefficients[i][j]>=0.f)
            coefficients[i][j]-=threshold;
          else
            coefficients[i][j]+=threshold;
        }
      }
    }

  return;
}

/*--------------------------------------------------------------------------*/
void garrote_shrinkage(int nx, int ny, float** coefficients, float threshold) {
  int i,j;
  int count = 0;

  if (threshold<0.000001f) {
    printf("Threshold too low for Garrote-shrinkage, aborting\n");
    return;
  }
  for(j=0; j<nx; j++)
    for(i=0; i<ny; i++) {
      /* Supplement your own code here */
      if (i>0 || j>0) {
        if (fabsf (coefficients[i][j]) <= threshold) {
          coefficients[i][j] = 0.0;
          count++;
        } else {
          coefficients[i][j]=coefficients[i][j]
            -(threshold*threshold)/coefficients[i][j];
        }
      }
    }

  return;
}
```

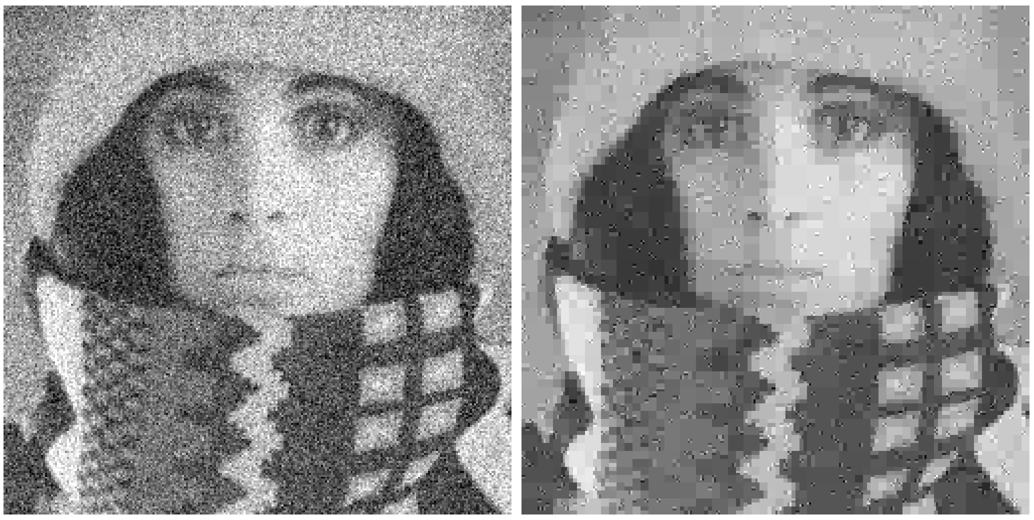

Figure 1: *Left:* Noisy version with Gaussian noise of standard deviation 30 (`trui-n30.pgm`). *Right:* Hard shrinkage with $T = 70$.

Figure 4 shows reasonable results for denoised images using Gaussian convolution and NL-means.

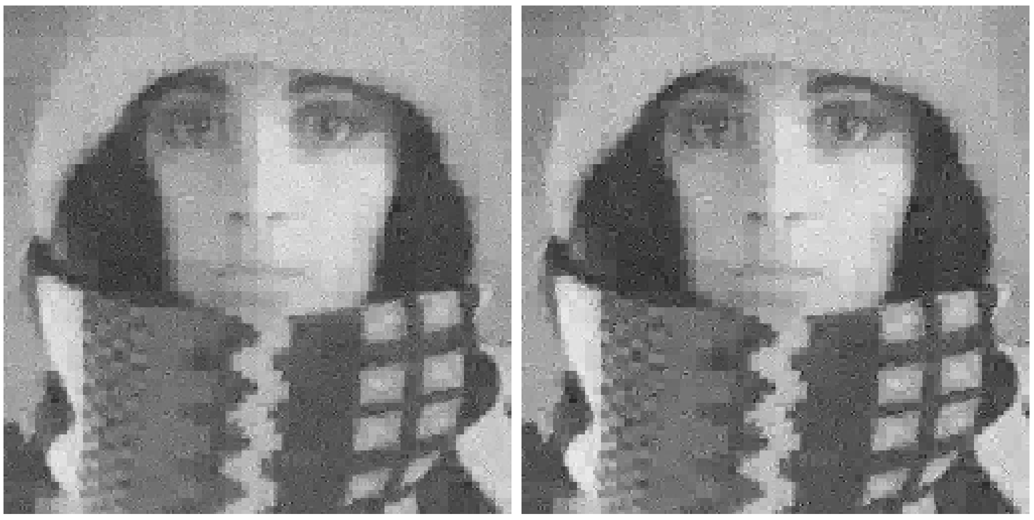

Figure 2: *Left:* Soft shrinkage with $T = 40$. *Right:* Garrote shrinkage with $T = 50$.

(b) Since hard shrinkage only modifies coefficients below the threshold T, a much larger threshold is needed in order to get rid of most of the noise. However, even at very large thresholds, e.g. $T = 70$, small, isolated parts of the noise remain clearly visible in the image (see Figure 1).

For soft and Garrote shrinkage, results of similar visual quality can be achieved for lower thresholds. One obvious difference is that just like for the hard shrinkage, some noisy parts of the image survive, but due to the rescaling of the coefficients, they don't stand out as much. Due to the different scaling functions of soft and Garrote shrinkage, Garrote shrinkage usually requires thresholds that are a bit larger than the ones for soft shrinkage (see Figure 2).

Compared to NL-means (Problem 4), wavelet shrinkage results are usually worse.

Problem 4 (NL-Means)

Given was the noisy image `trui-n30.pgm` as depicted in Figure 3, right side. It was created by adding Gaussian noise of standard deviation 30 to the original image, given on the left side.

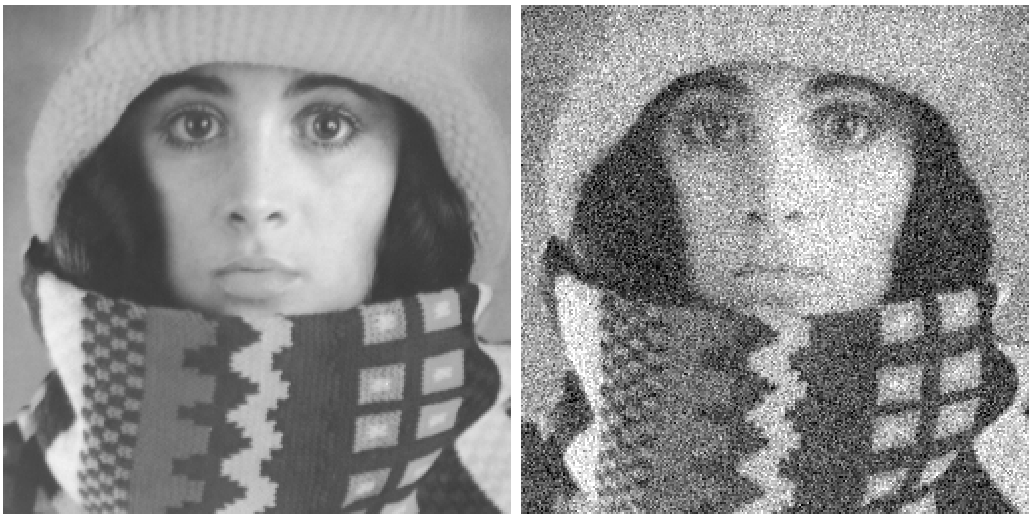

Figure 3: *Left:* Original image. *Right:* Noisy version with Gaussian noise of standard deviation 30 (`trui-n30.pgm`).

(a)+(b) Figure 4 shows reasonable results for denoised images using Gaussian convolution and NL-means.

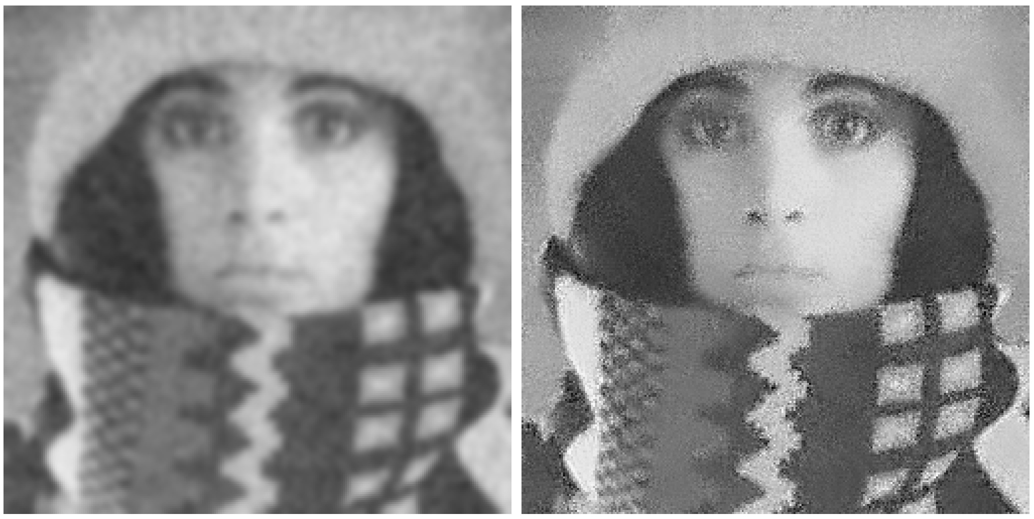

Figure 4: Denoised versions of `trui-n30.pgm` using Gaussian convolution with standard deviation $\sigma = 2$ (*left*) and NL-means with patch radius $m = 4$, search-window-size $n = 10$ and filter parameter $\sigma = 130$ (*right*).

(c) The best way to find optimal parameters for any denoising method and to judge the quality of denoised results is to compare the result with the original image. But actually the original image is usually not given because denoising would then be senseless. In contrast the method noise image can be computed by just using the filtered image and the given noisy image. Since method noise is the difference of these two images, it shows which information was removed by a filter. In the optimal case we expect to see noise, in our case Gaussian noise of standard deviation 30.

The noise consists of positive as well as negative values. Thus we have to shift the result by 127.5 such that we get only positive values, which can then be visualised.

Figure 5, shows the method noise for the filtered version of `trui-n30.pgm` (cf. Figure 4, right side).

We clearly see, that in both cases not only noise, but also structures of textures and image details are removed. However, these structures are much more present in the case of Gaussian convolution. There, the shape of Lena is clearly visible and in particular, high contrast edges have been removed. Furthermore, the texture of the feathers is much less pronounced in the method noise for NL-Means.

So NL-means is a much more appropriate method for denoising than

Gaussian convolution. Indeed it is one of the best methods available for denoising so far. This estimation is also confirmed when comparing the results shown in Figure 4 with the original image given in Figure 3. Please note that we use the simplest possible version of the NL-means filter in this exercise. There are many variations and extensions to this concept that yield even better results.

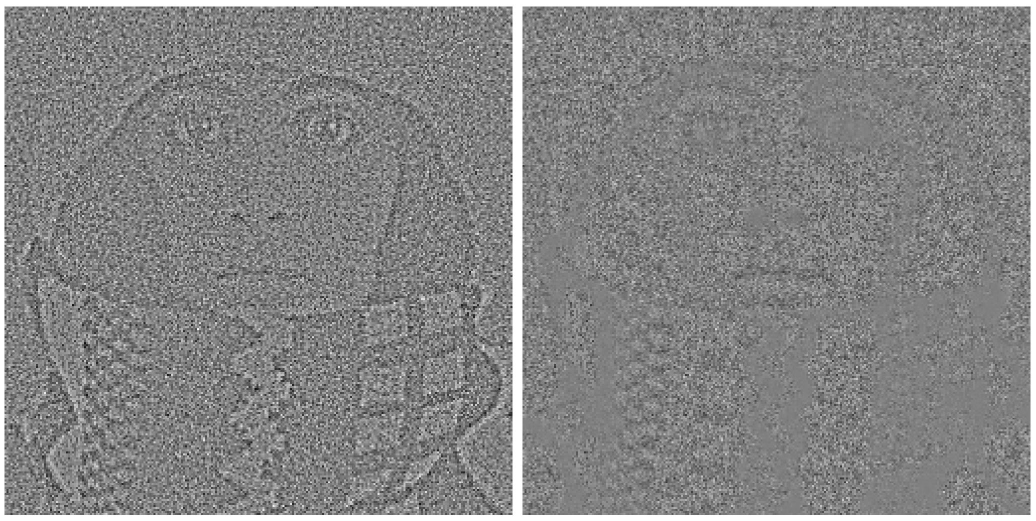

Figure 5: Method noise with respect to the images of Figure 4. *Left:* Method noise w.r.t. Gaussian Convolution. *Right:* Method noise w.r.t. NL-means.