Example Solutions for Homework Assignment 1 (H1)

Problem 1 (Continuous Convolution)

For the given convolution kernel f and any continuous signal $g(x)$ we have

$$(g * f)(x) = \int_{-\infty}^{\infty} f(x-z)\, g(z)\, \mathrm{d}z \;= \int_{x-1}^{x+1} \frac{1}{2}\, g(z)\, \mathrm{d}z\,.$$

Note that due to commutativity, it doesn't matter in which order f and g appear in the convolution term.

For $f * f$ we therefore obtain

$$\begin{aligned}(f * f)(x) &= \int_{x-1}^{x+1} \frac{1}{2}\, f(z)\, \mathrm{d}z \\ &= \begin{cases} 0, & x \le -2, \\ \int_{-1}^{x+1} \frac{1}{4}\, \mathrm{d}z, & -2 < x \le 0, \\ \int_{x-1}^{1} \frac{1}{4}\, \mathrm{d}z, & 0 < x \le 2, \\ 0, & x > 2 \end{cases}\end{aligned}$$

and thus

$$(f * f)(x) = \begin{cases} 0, & x \le -2, \\ \frac{2+x}{4}, & -2 < x \le 0, \\ \frac{2-x}{4}, & 0 < x \le 2, \\ 0, & x > 2\,. \end{cases}$$

For $f * f * f$ we find

$$
\begin{aligned}
(f * f * f)(x) &= \int_{x-1}^{x+1} \frac{1}{2}\,(f * f)(z)\,\mathrm{d}z \\
&= \begin{cases}
0, & x \le -3, \\
\int_{-2}^{x+1} \frac{2+z}{8}\,\mathrm{d}z, & -3 < x \le -1, \\
\int_{x-1}^{0} \frac{2+z}{8}\,\mathrm{d}z + \int_{0}^{x+1} \frac{2-z}{8}\,\mathrm{d}z, & -1 < x \le 1, \\
\int_{x-1}^{2} \frac{2-z}{8}\,\mathrm{d}z, & 1 < x \le 3, \\
0, & x > 3\,;
\end{cases} \\
(f * f * f)(x) &= \begin{cases}
0, & x \le -3, \\
\frac{x^2+6x+9}{16}, & -3 < x \le -1, \\
\frac{6-2x^2}{16}, & -1 < x \le 1, \\
\frac{x^2-6x+9}{16}, & 1 < x \le 3, \\
0, & x > 3\,.
\end{cases}
\end{aligned}
$$

If we evaluate $f * f$ at the indicated positions, we obtain:

$$
\begin{aligned}
(f * f)(-1) &= \frac{1}{4} \\
(f * f)(0) &= \frac{1}{2} \\
(f * f)(1) &= \frac{1}{4} \\
(f * f)(2) &= 0 \qquad (f * f)(3) \qquad = 0
\end{aligned}
$$

which corresponds to the discrete convolution $(f * f)$ from the classroom assignment. As for $(f * f * f)$ we get

$$
\begin{aligned}
(f * f * f)(-2) &= \frac{1}{16} \\
(f * f * f)(-1) &= \frac{1}{4} \\
(f * f * f)(0) &= \frac{6}{16} \\
(f * f * f)(1) &= \frac{1}{4} \\
(f * f * f)(2) &= \frac{1}{16} \\
(f * f * f)(3) &= 0 \\
(f * f * f)(4) &= 0
\end{aligned}
$$

which corresponds to the discrete convolution $(f*f*f*f)$ from the classroom assignment. We note here that the correspondences are not exact, in particular the indices of the spatial grid points are not aligned and corresponding results do not necessarily take the same number of iterations. Nevertheless, both the continuous and the discrete convolutions of box filters show the same tendencies. The common patterns $\frac{1}{4}, \frac{1}{2}, \frac{1}{4}$ and $\frac{1}{16}, \frac{4}{16}, \frac{6}{16}, \frac{4}{16}, \frac{1}{16}$ are actually so called binomial kernels which are used to approximate Gaussian filters and will be discussed in upcoming lectures. In particular, both the continuous and discrete convolutions of box filters approximate Gaussians better and better the higher the number of convolutions.

Problem 2 (Properties of the Convolution)

The continuous convolution between f and g is given by

$$(f * g)(x) = \int_{\mathbb{R}} f(t)g(x-t)\mathrm{d}t$$

It follows,

$$\begin{aligned}
(f * g)(x) &= \int_{-\infty}^{\infty} f(t)g(x-t)\mathrm{d}t \\
&= \int_{\infty}^{-\infty} f(x-z)g(x-(x-z))(-1)\mathrm{d}z \\
&= \int_{-\infty}^{\infty} g(z)f(x-z)\mathrm{d}z \\
&= (g * f)(x)
\end{aligned}$$

and thus the commutativity holds. As for the associativity, it is easy to see that

$$\begin{aligned}
((f * g) * h)(x) &= \int_{\mathbb{R}} (f * g)(z)h(x-z)\mathrm{d}z \\
&= \int_{\mathbb{R}}\int_{\mathbb{R}} f(t)g(z-t)h(x-z)\mathrm{d}t\mathrm{d}z \quad y \leftrightarrow x-z \\
&= \int_{\mathbb{R}}\int_{\mathbb{R}} f(t)g(x-t-y)h(y)\mathrm{d}t\mathrm{d}y \\
(f * (g * h))(x) &= \int_{\mathbb{R}} f(t)(g * h)(x-t)\mathrm{d}t \\
&= \int_{\mathbb{R}} f(t)(h * g)(x-t)\mathrm{d}t \\
&= \int_{\mathbb{R}}\int_{\mathbb{R}} f(t)h(z)g(x-t-z)\mathrm{d}t\mathrm{d}z
\end{aligned}$$

The result follows now immediately after renaming the variable y back into z. The distributivity follows almost immediately from the definition of the convolution

$$\begin{aligned}(f+g)*h(x) &= \int_{\mathbb{R}} (f+g)(t)h(x-t)\mathrm{d}t \\ &= \int_{\mathbb{R}} f(t)h(x-t)\mathrm{d}t + \int_{\mathbb{R}} g(t)h(x-t)\mathrm{d}t \\ &= f*h(x) + g*h(x)\end{aligned}$$

The second equations is now a direct consequence of the first one combined with the commutativity.

We compute the derivative $(g*f)'$

$$\begin{aligned}(g*f)'(x) &= \frac{d}{dx}\int_{-\infty}^{\infty} g(x-y)\ f(y)\ dy \\ &= \int_{-\infty}^{\infty} \frac{d}{dx} g(x-y)\ f(y)\ dy \\ &= \int_{-\infty}^{\infty} g'(x-y)\ f(y)\ dy \\ &= (g'*f)(x)\end{aligned}$$

We have assumed that all assumptions necessary or exchanging the derivative and the integral are satisfied. Since g is n times differentiable, we can repeat this step and write

$$\frac{d^n}{dx^n}(g*f) = \left(\frac{d^n}{dx^n}g\right) * f$$

We have shown that the n-th derivative of $(g*f)$ exists.

For the n-th derivative of $(f*g)$, it suffices to use the commutativity. The argumentation is the same in that case.

Problem 3 (Noise, Quantisation and Dithering)

First of all we consider the code that has to be supplemented for each subproblem:

(a) Expected supplemented code for the required quantisation:

```
/* ---- process image ---- */
d = powf(2.0,8-q);
for (j=1; j<=ny; j++){
  for (i=1; i<=nx; i++){
    u[i][j] = (floorf(u[i][j] / d) + 0.5) * d;
  }
}
```

(b) Expected supplemented code for the required quantisation with noise:

```
/* ---- process image ---- */
d = powf(2.0,8-q);
double n = 0.0;
for (j=1; j<=ny; j++){
  for (i=1; i<=nx; i++){
    n = -0.5 + rand()/RAND_MAX;
    u[i][j] = (floorf(u[i][j] / d + n) + 0.5) * d;
  }
}
```

(c) Expected supplemented code for the required Floyd-Steinberg algorithm:

```
/* ---- process image ---- */
float d = powf(2.0f,8-q);
float old = 0;
float error = 0;
for (j=1; j<=ny; j++){
  for (i=1; i<=nx; i++){
    old = u[i][j];
    u[i][j] = (floorf(u[i][j] / d) + 0.5)*d;
    error = (old-u[i][j]);
    u[i+1][j ] += 7.0/16.0*error;
    u[i-1][j+1] += 3.0/16.0*error;
    u[i ][j+1] += 5.0/16.0*error;
    u[i+1][j+1] += 1.0/16.0*error;
  }
}
```

As we can see in Tab. 1, there is almost no visible difference between the original image and the one with $q = 6$, (e.g. 64 colours). This confirms the statement from the lecture

that the human eye cannot distinguish many different grey values.

For $q = 3$ we clearly see a loss in quality compared to the original image. Especially in regions with a smooth gradient (such as the clouds) the reduction in the number of colours really deteriorates the image. The quantisation without noise creates clear discontinuities and sharp edges. These discontinuities are further enhanced by the so called lateral inhibition in our eye so that the visual quality of the image suffers a lot.

Adding some noise, as done in the second column reduces this effect. In areas where a certain pixel value is on the edge between two different quantisation levels, the noise forces some of the pixel on the one side and some on the other such that an overall smoother transition between the two areas is achieved. This effect is especially well on the lower part of the boat in the centre of the picture.

However, the best results are obtained through the Floyd-Steinberg Algorithm. Similarly as in (b), we get smooth transitions (e.g. on the clouds and the lower part of the boat). Our visual system does not consider each pixel individually, but instead it does some smoothing, such that the perception in a given point is also dependent on its neighbourhood. The Floyd-Steinberg algorithm exploits this fact to redistribute the quantised values to the neighbourhood such that our brain perceives an image that looks similar to the original one.

Using $q = 1$ reveals how the current implementation treats over- and undershoots. The noisy method as well as the Floyd-Steinberg algorithm yield images with more than two colours. The reason for this lies in the fact that the noise can lead to pixels with values larger than 255 and smaller than 0. These values are mapped to 0 (resp. 255) when the file is written to disk. Thus we actually obtain images with 4 colours, namely 0,64,192 and 255. In order to avoid this effect one could also truncate over- and undershoots, i.e. one would map the value 0 to 64 and 255 to 192. However, this would change final visual result only slightly.

Original Image

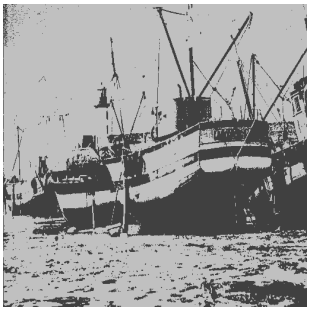
	Without noise	With noise	Floyd Steinberg
$q = 6$			
$q = 3$			
$q = 1$			

Table 1: Results for the different quantisation algorithms and varying numbers of colors used.