Image Processing and Computer Vision (IPCV)

Prof. Dr. J. Weickert
Mathematical Image Analysis Group

Winter term 13/14
Saarland University

Example Solutions for Classroom Assignment 1 (C1)

Problem 1 (Discrete Convolution)

We observe that with the given convolution kernel f one obtains for any discrete signal $g = (g_i)_{i \in \mathbb{Z}}$ the result

$$(g * f)_i = \sum_{j=-\infty}^{\infty} f_{i-j}\, g_j = \frac{1}{2}\, g_i + \frac{1}{2}\, g_{i-1}\ .$$

Applying this iteratively leads to

$$(f * f)_i = \begin{cases} 0, & i < 0 \text{ or } i > 2, \\ \frac{1}{4}, & i = 0, 2, \\ \frac{2}{4}, & i = 1 \end{cases}$$

$$(f * f * f)_i = \begin{cases} 0, & i < 0 \text{ or } i > 3, \\ \frac{1}{8}, & i = 0, 3, \\ \frac{3}{8}, & i = 1, 2 \end{cases}$$

$$(f * f * f * f)_i = \begin{cases} 0, & i < 0 \text{ or } i > 4, \\ \frac{1}{16}, & i = 0, 4, \\ \frac{4}{16}, & i = 1, 3, \\ \frac{6}{16}, & i = 2\ . \end{cases}$$

Problem 2 (Continuous Convolution)

We have

$$(k * h)(x) = \int_{-\infty}^{\infty} k(t) h(x - t) \mathrm{d}t = \int_{-\infty}^{x} k(t) \mathrm{d}t$$

For $x < -1$ this integral is 0 as the function k itself is always 0 for $t < -1$. For $x \geqslant 1$ we obtain the full integral

$$(k * h)(x) = \int_{-1}^{1} k(t) \mathrm{d}t = 1$$

The case $-1 \leqslant x < 0$ we obtain

$$(k * h)(x) = \int_{-1}^{x} 1 + t \mathrm{d}t = \frac{1}{2}\, (1 + x)^2$$

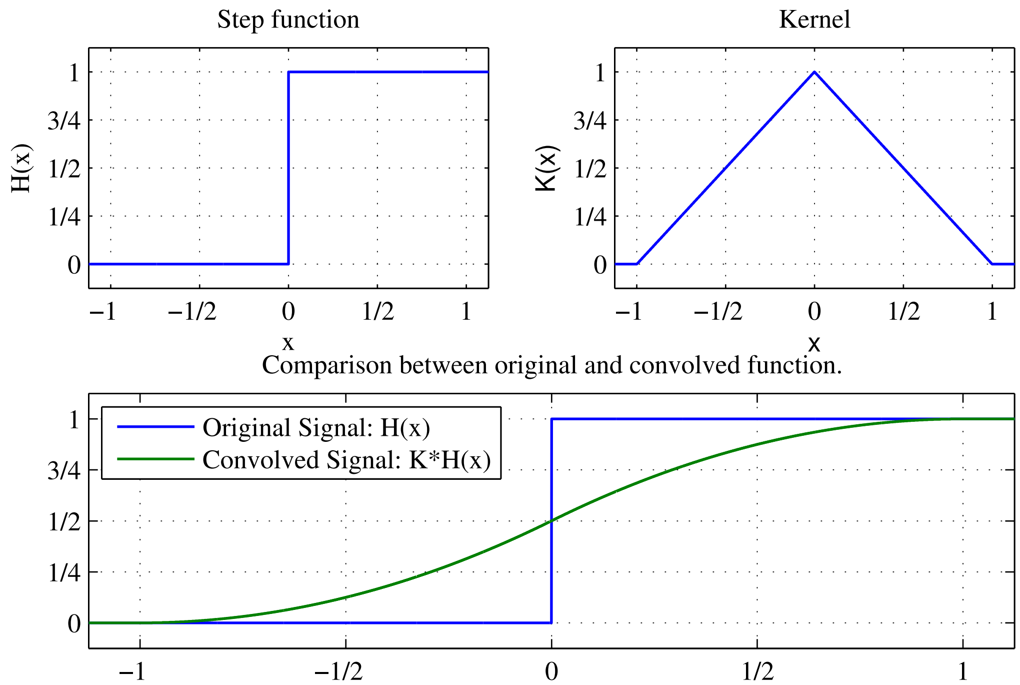

Figure 1: The function h and its convolution with kernel k.

Finally, for $0 \leqslant x < 1$ we obtain

$$(k * h)(x) = \int_{-1}^{0} 1 + t\mathrm{d}t + \int_{0}^{x} 1 - t\mathrm{d}t = \frac{1}{2} + x - \frac{1}{2}x^2$$

This means that we obtain

$$(k * h)\,(x) = \begin{cases} 0, & x < -1 \\ \frac{1}{2}\,(1+x)^2\,, & -1 \leqslant x < 0 \\ \frac{1}{2} + x - \frac{1}{2}x^2, & 0 \leqslant x < 1 \\ 1, & x \geqslant 1 \end{cases}$$

As we can see in Fig. 1 the convolution with k resulted in a much smoother signal. The sharp boundary present in h has completely vanished.

Problem 3 (Box-Muller Algorithm)

The following theorem shows how one can use uniformly distributed random numbers to generate random variables obeying any distribution. The Box-Muller Algorithm is basically just a clever application of this theorem.

Transformation/Inversion Rule: *Let F be a continuous distribution function on $\mathbb{R}$ with inverse F^{-1}. If U is a uniform $[0, 1]$ distributed random variate, then the random variate $X = F^{-1}(U)$ has distribution function F. Conversely, if X has distribution function F, then $F(X)$ is uniformly distributed on $[0, 1]$.*

The proof of this theorem is quite simple, however, it requires elementary knowledge in measure and probability theory and thus we skip it at this place and refer instead to the excellent Book of Luc Devroye, *Non-Uniform Random Variate Generation*[1], which describes the complete theory of random number generation in every detail.

Note that the distribution function of a random variable X is defined in terms of probability and probability density function as

$$F_X(x) := P\left(X \leqslant x\right) := \int_{-\infty}^{x} p(t)\,\mathrm{d}t$$

From this it also follows that

$$\frac{\partial}{\partial x} F_X\left(x\right) = p(x)$$

although we will not need that relation in this exercise. The above theorem essentially states that it is enough to find an expression for F^{-1}. If an evaluation of $F^{-1}(U)$ yields the values stated in the Box-Muller algorithm, we have proven that it generates normally distributed random numbers. Note that certain fundamental theorems from measure and probability theory guarantee that this inverse function always exists and is unique. However, it may not always have an analytical description. The one dimensional normal distribution function for example is given by

$$F(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right) \mathrm{d}t = \frac{1}{2}\left(1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right)$$

where erf is the Gaussian error function, which cannot be described analytically. Further, its inverse cannot be described in a closed form either. Thus, a straightforward application of the above theorem is not possible. One could of course use numerical methods to invert the distribution function, however, it would be computationally too expensive to deploy such an approach on a large scale. In 1958, George Edward Pelham Box and Mervin Edgar Muller suggested to consider the normal distribution in $\mathbb{R}^2$ and to generate pairs of normal distributed random numbers instead of single ones.[2]

The normal distribution in $\mathbb{R}^2$ has the density function

$$p(x,y) = \frac{1}{2\pi\sigma^2} \exp\left(\frac{-(x-\mu_1)^2 - (y-\mu_2)^2}{2\sigma^2}\right)$$

For the standard normal distribution, i.e. $\mu = 0$ and $\sigma = 1$, this simplifies to:

$$p(x,y) = \frac{1}{2\pi} \exp\left(-\frac{x^2+y^2}{2}\right) \tag{1}$$

Note that this function only depends on the distance to the origin and not on the direction. This little fact will be crucial in the following. We now need two further results from probability theory.

[1] Available for free at `http://cg.scs.carleton.ca/~luc/rnbookindex.html`

[2] A Note on the Generation of Random Normal Deviates. The Annals of Mathematical Statistics (1958), Vol. 29, No. 2 pp. 610–611.

Theorem *If two random variates X and Y are independent, their common probability density function is the product of the individual probability density functions.*

$$p_{(X,Y)}(x,y) = p_X(x)p_Y(y)$$

Theorem *If the random variate pairs (X,Y) and (M,N) have probability density functions $p_{(X,Y)}$ and $p_{(M,N)}$, and if there exists a C^1 diffeomorphism g with Jacobian determinant different from 0, then the probability density functions are linked by the relation*

$$p_{(M,N)} = |\det(\boldsymbol{J}_g)|p_{(X,Y)}$$

We now want to apply the two previous theorems and chose as diffeomorphism the polar coordinate transform (polar coordinates are always useful for functions that only depend on the distance from the origin):

$$\begin{aligned} x &= r\cos\varphi =: g_1(r,\varphi) \\ y &= r\sin\varphi =: g_2(r,\varphi). \end{aligned}$$

The Jacobian of the polar coordinate transformation can be computed as

$$\boldsymbol{J}_g(r,\varphi) = \begin{pmatrix} \dfrac{\partial g_1}{\partial r} & \dfrac{\partial g_1}{\partial \varphi} \\ \dfrac{\partial g_2}{\partial r} & \dfrac{\partial g_2}{\partial \varphi} \end{pmatrix} = \begin{pmatrix} \cos(\varphi) & -r\sin\varphi \\ \sin(\varphi) & r\cos\varphi \end{pmatrix}$$

and its determinant is given by

$$\det(\boldsymbol{J}_g(r,\varphi)) = r\cos^2\varphi + r\sin^2\varphi = r$$

From this it follows that for independent normal distributed random variates X and Y we have

$$p_{(M,N)}(r,\varphi) = p_{(X,Y)}(x,y)\det(\boldsymbol{J}_g) = p_X(x)p_Y(y)\det(\boldsymbol{J}_g) = \exp(-\frac{r^2}{2})\frac{1}{2\pi}r$$

where the random variates M and N are the corresponding representations in polar coordinates. If the pair (X,Y) of random variates is independent, the same also holds for the pair $(M,N) := (g_1^{-1}(X,Y), g_2^{-1}(X,Y))$. Thus M must have the density $r\exp\left(-\frac{r^2}{2}\right)$ (with domain $[0,\infty)$) and N has the density $\frac{1}{2\pi}$ on $[0,2\pi)$. This means, that at this point we have reduced our initial problem down to generating random variates M and N of the latter two distributions and transforming the obtained numbers back into cartesian coordinates. Since N is uniformly distributed on $[0,2\pi)$, it suffices to generate uniformly distributed numbers on $[0,1]$ and to multiply them by 2π. As for M we may apply our first theorem. We have

$$F(t) = \int_0^t r\exp(-\frac{r^2}{2})\mathrm{d}r = 1 - \exp(-\frac{t^2}{2})$$

Now, solving the equation $F(M) = U$ is quite easy. Its solution is given by

$$M = \sqrt{-2\ln(1-U)}$$

Since U is uniformly distributed on $[0,1]$, the same also holds for $1-U$ and thus we can further write $M = \sqrt{-2\ln(U)}$. Finally, this means that we have the following algorithm:

1. Generate U, V uniformly distributed on $[0,1]$.

2. Compute

$$\begin{aligned} M &= \sqrt{-2\ln(U)} \\ N &= 2\pi V \end{aligned}$$

3. Use the polar coordinate transform to go back to cartesian coordinates

$$\begin{aligned} X &= M\cos(N) \\ Y &= M\sin(N) \end{aligned}$$

Remark: For random variables $\tilde{X}, \tilde{Y}$ of a general normal distribution with arbitrary σ and μ we compute

$$\begin{aligned} \tilde{X} &= \mu + \sigma X \\ \tilde{Y} &= \mu + \sigma Y \end{aligned}$$

Remark: The Box-Muller algorithm is not the only way to generate normal distributed numbers. It was criticised in the past because it required the computation of trigonometric functions as well as a logarithm, which used to be ($\geqslant$ 30 years ago) computationally intensive on computers. Therefore, people suggested alternatives like the algorithm of Marsaglia [3]. Nowadays, most good random number generators use the so called Ziggurat Method [4].

[3] A convenient method for generating normal variables, G. Marsaglia and T. A. Bray, SIAM Rev. 6, 260-264, 1964

[4] The Ziggurat Method for Generating Random Variables, G. Marsaglia and W. W. Tsang, Journal of Statistical Software 5 (8), 2000