Image Analysis Excercise Sheet 3

Markus Doering, 3153320

May 8, 2013

1 Convolution in 1-D

Given are an input signal g and a filter h, which will act on the signal:

$$g(x) = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}^{\mathrm{T}}$$
$$h(x) = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 & 0 & 0 \end{pmatrix}^{\mathrm{T}}.$$

The output signal will be

$$(g * h)(x) = \begin{pmatrix} 4 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 3 \end{pmatrix}^{\mathrm{T}},$$

a smoothened, shifted and amplified version of the signal. The circulant matrix of the filter is given by

$$H = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 2 \\ 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 2 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

The convolution of g and h can be computed as

$$g * h = H \cdot g.$$

Theorem. *Let g be a signal and let h be a filter with*

$$\sum_k (h * g)(k) = \sum_k g(k).$$

In the general case $\sum_k g(k) \neq 0$, we find that the DC term $\sum_k h(k) = 1$.

Proof. Define $s := (1, \dots, 1)^{\mathrm{T}}$.

$$\begin{aligned}\sum_j g(j) &= \sum_k (h * g)(j) = s^{\mathrm{T}} (h * g) \overset{\text{Convolution Theorem}}{=} \\ &= s^{\mathrm{T}} \mathcal{F}^{-1} \left(\mathcal{F} f \odot \mathcal{F} g \right) = s^{\mathrm{T}} F^{\dagger} \left(F f \odot F g \right) \frac{1}{n} = \\ &= (1, 0, 0, \dots, 0) \left(F f \odot F g \right) = \left(s^{\mathrm{T}} f \cdot s^{\mathrm{T}} g \right) = \\ &= \left(\sum_k h(k) \right) \left(\sum_j g(j) \right)\end{aligned}$$

We can divide both sides by the signal's sum and get $\sum_k h(k) = 1$. □

2 DFT in 1-D

Given is the transformed function

$$\mathring{g}(k) = \sqrt{n} \frac{i}{2} \left(\delta_{+2}(k) - \delta_{-2}(k) \right),$$

and we want to compute the original function by discrete Fourier transform. Let $F_{l,\cdot}$ denote the l-th row of the Fourier matrix.

$$\begin{aligned} g(l) &= \frac{1}{\sqrt{n}} F_{l,\cdot} \mathring{g} = \frac{i}{2} \sum_{k=0}^{n-1} e^{\frac{2\pi i}{n} lk} \left(\delta_{+2}(k) - \delta_{-2}(k) \right) \\ &= \frac{i}{2} \left(e^{\frac{4\pi i}{n} l} - e^{-\frac{4\pi i}{n} l} \right) = \frac{i}{2} \left(2i \sin \left(\frac{4\pi}{n} l \right) \right) = -\sin \left(\frac{4\pi}{n} l \right) \end{aligned}$$

3 DFT in 2-D

When two matrices have no spectral overlap, the convolution theorem asserts that

$$a * b = \mathcal{F}^{-1} \left(\mathcal{F}(a) \odot \mathcal{F}(b) \right) = \mathcal{F}^{-1}(0) = 0.$$

The python code is found in the appendix.

4 Correlation Theorem

Theorem. *Cross Correlation Theorem*

$$\mathcal{F} \left(a \star g \right) = \mathcal{F}(a)^* \odot \mathcal{F}(g)$$

Proof. We observe that the cross correlation operation can be stated in terms of the circulant matrix A of a as

$$a \star g = A^{\dagger} g.$$

We recall the equation

$$AW = W\mathring{A} \;\Leftrightarrow\; W^\dagger A^\dagger = \mathring{A}^\dagger W^\dagger \;\Leftrightarrow\; W^\dagger A^\dagger W = \mathring{A}^*,$$

where $\mathring{A}$ is the diagonal matrix with entries $\mathring{a} = \mathcal{F}a$ and W is the normalized Fourier matrix ($WW^\dagger = I$). We obtain

$$\begin{aligned}\mathcal{F}\left(a \star g\right) &= \mathcal{F}\left(A^\dagger g\right) = W^\dagger A^\dagger g = W^\dagger A^\dagger W W^\dagger g \\ &= \mathring{A}^* W^\dagger g = \mathcal{F}(a)^* \odot \mathcal{F}(g).\end{aligned}$$

□

```python
#/usr/bin/python
# coding: utf-8
#
# Author: Markus Doering
# File: ia_03_03.py
#

from numpy import set_printoptions, zeros, finfo, \
    exp, pi, isreal, real_if_close, fliplr, flipud
from numpy.random import rand
from numpy.fft import fftshift, fft2, ifft2, ifftshift
from scipy.signal import fftconvolve, convolve2d
set_printoptions(precision=2, linewidth=180)

import matplotlib
matplotlib.use('Qt4Agg')
from matplotlib import pyplot as plot

from pprint import pprint as pp


def getmat(width_low, width_med, n, content='low'):
    assert n%2 == 1, "n must be an odd number"
    assert width_low>=0, "width must be nonnegative"
    assert width_med>=0, "width must be nonnegative"

    mid = n/2

    # inner rectangle mask
    low = zeros((n,n))
    low[mid-width_low:mid+width_low+1,\
        mid-width_low:mid+width_low+1] = 1

    # middle frame mask
    med = zeros((n,n))
    med[mid-width_low-width_med:mid+width_low+width_med+1,\
        mid-width_low-width_med:mid+width_low+width_med+1] = 1
    med = med * (1-low)

    # random complex numbers of size nxn with amplitude 1
    rnd = exp(2*pi*1j*rand(n,n))

    if content=="low":
        mat = low*rnd
```

```python
    else:
        mat = med*rnd

    # make ifft real
    mat = (mat + fliplr(flipud(mat)).conjugate())/2

    mat = ifftshift(mat)

    # check if we did it right
    should_be_real = real_if_close(ifft2(mat))
    assert isreal(should_be_real).all(), \
        "The inverse FFT was not real."

    return mat


if __name__ == "__main__":

    a = getmat(2,2,51,content='low')
    b = getmat(2,2,51,content='med')

    A = real_if_close(ifft2(a))
    B = real_if_close(ifft2(b))

    plot.figure()
    plot.subplot(1,2,1)
    plot.imshow(A, interpolation='nearest')
    plot.gray()

    plot.subplot(1,2,2)
    plot.imshow(B, interpolation='nearest')

    plot.show()

    C = convolve2d(A,B,boundary='wrap')

    if (abs(C) < finfo(float).eps).all():
        print("The images convolved to zero.")
    else:
        print("Convolution did not return zero.")
```