Chapter 1

Introduction

This lecture is about basic concepts of digital image processing. As it is part of a course in environmental physics, the focus of mostly on topics relevant for this domain, in particular remote sensing.

1.1 Digital Images

- In science, we often have to deal with multidimensional data
- Typical source of data: measurements of a physical parameter (temperature, air pressure, radiance etc.)
- often multidimensional (2 dim. and more), e.g.:
  - many point measurements of air pressure in a region
  - any satellite image
- How to interpret/analyse such data?
  1. visualise them, i.e., display as diagram or *image*
  2. later: further analysis, computer-aided, automatic (?)

$\Rightarrow$ basically we have to apply image processing techniques

Image: 2D function $f(x,y)$ defined on some region $\{x,y \in \mathbb{R} | 0 \leq x \leq X, 0 \leq y \leq Y\}$

`Figure: rectangle XY`

- x,y, and image value $f(x,y)$ continuous (real number)
- $f(x,y)$ is something like the image brightness at position (x,y)

Digital image: 2D array (matrix) A_{mn} of digital numbers, and $\{m,n \in \mathbb{N} | 0 \leq m \leq M, 0 \leq n \leq M\}$

`Figure: rectangle MN`

- m, n, and the image value A_{mn} are discrete

- only a finite number of possible values (digital numbers!)
- typical: number of possible values is a power of 2, e.g. $0 \ldots 255$: 2^8 different values, “8 bit quantisation”
- when displayed, the numbers are typically assigned a grey level (black, shades of grey, white), so each number is then one dot (*pixel*, from “picture element”) in a 2-D rectangular image.
- typical (but just convention): 0: black, highest possible value: white

- Why *digital* images? – Technology (computer)
- “Reality”: Analogue image (continuous range of values) $\xrightarrow{\text{digitisation}}$ Digital image: discrete range of values
- Nowadays most sensors immediately digitise and output digital data to the user[1]

⇒ we might have to consider what digitisation has done to the originally continuous measured parameters

→Fig. 1.1

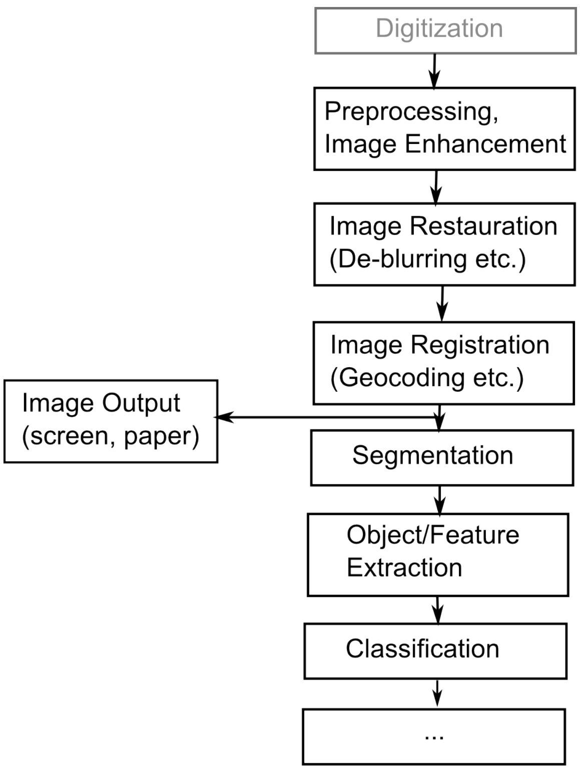

Fig. 1.1: *Flowchart of “classical” image processing steps*

[1]see, e.g., VIIRS on satellite Suomi-NPP: CCD produces analog signal which is then digitised on-board. See also `http://npp.gsfc.nasa.gov/science/sciencedocuments/082012/474-00027_ATBD-VIIRS-RadiometricCal_B_20120411.pdf`, p. 28, Fig. 13

1.2 Grey-level Histogram

- Simple tool to investigate some basic properties of a digital image
- A diagram showing the frequency of occurrence (y-axis) of each possible image value (grey level, x-axis)

  →Fig. 1.2

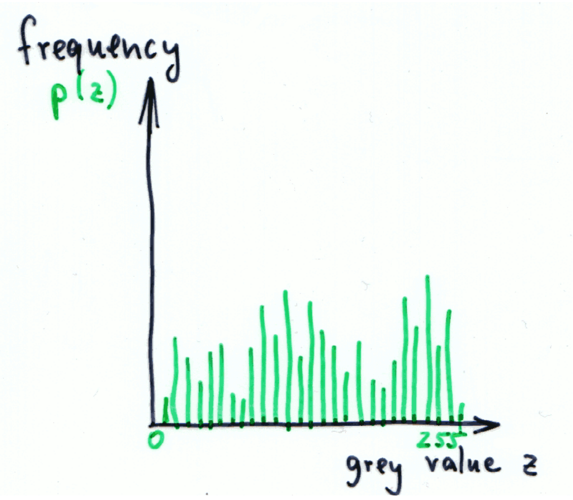

Fig. 1.2: *A histogram*

- frequency of occurrence either given as percentage of all pixels or as absolute number (scale often not essential)
- Shows, e.g., if an image is dark, or bright, if it uses the full range of possible grey levels, if the contrast is high or low. →Fig. 1.3
- Can also be used to distinguish objects and background, provided they have different grey level

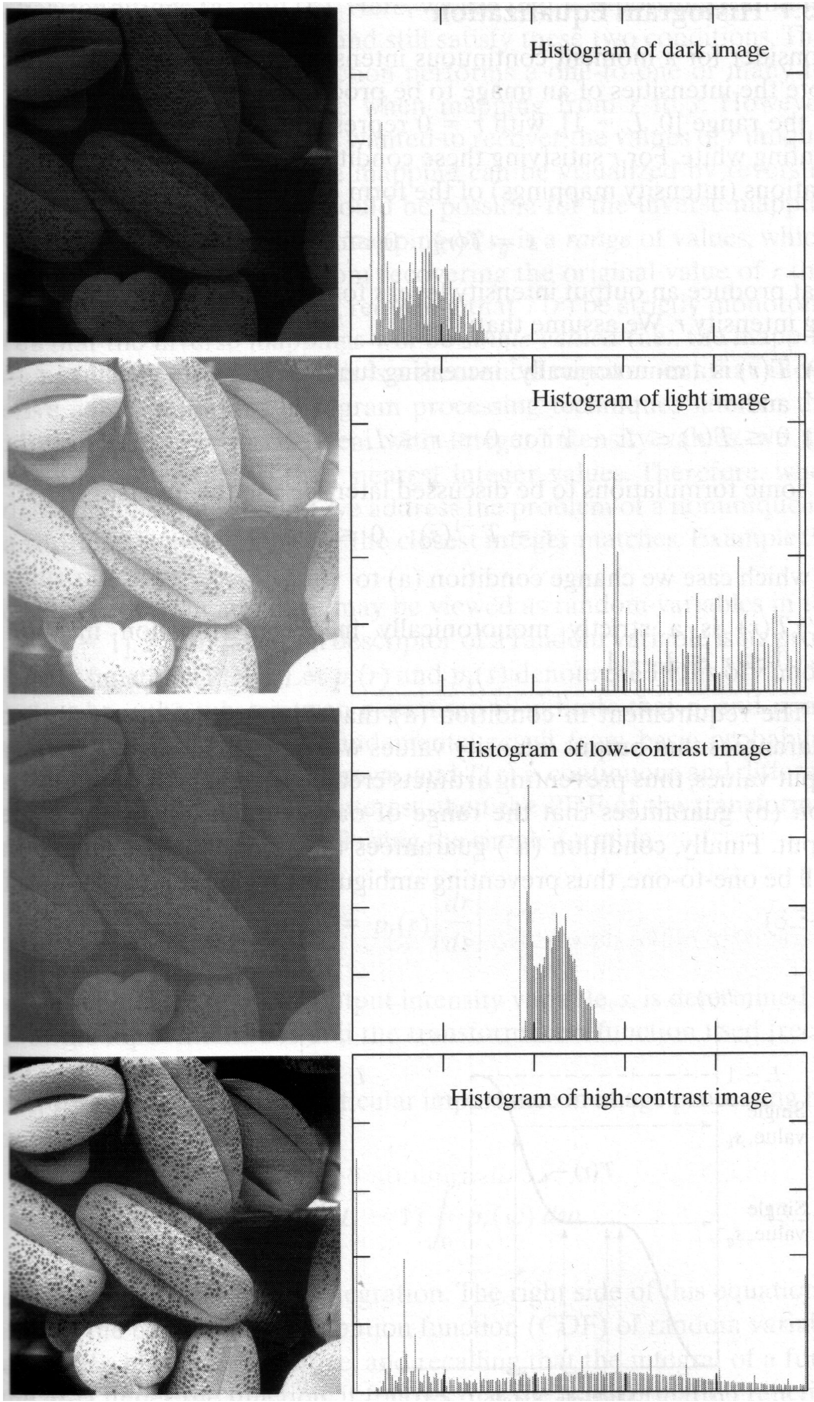

Fig. 1.3: *Four typical histograms and the corresponding images (Fig. 3.16 from Gonzalez and Woods, 2002)*

1.3 Operations on digital images

- Digital image processing = operations on matrices (arrays)
- three types of operations:

point operations: value of the new (output) pixel B_{mn} depends only on value of old (input) pixel A_{mn}

local operations: value of the new (output) pixel B_{mn} depends on a group of pixels (a *neighbourhood*) in input image A (Example: moving average)

global operations: value of the new (output) pixel B_{mn} depends on all pixels of input image A (Example: 2-dim. Fourier transformation of an image)

→Fig. 1.4

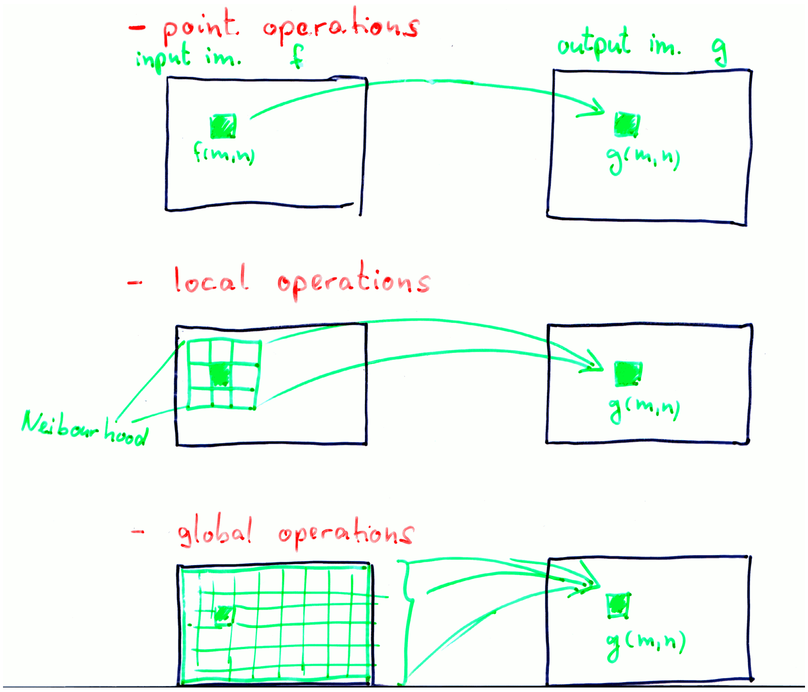

Fig. 1.4: *Three types of operations; here, f is the input image and g the output image.*

Bibliography

R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley, second edition, 2002.

Chapter 2

Image Enhancement

2.1 Grey-level transformations

- Grey-level transformation: point operation, modifying the pixel values
- map input image A to output image B:

  $$A \to B$$

  by transforming the pixel value (grey level) D_A of the input image into new pixel value (grey level) D_B:

  $$D_A \to D_B = f(D_A)$$

  where f can be any function.
- mainly used to enhance the contrast of an image:

Example 1: • in A, only grey levels between 0 and 128 occur

`→Fig. 2.1`

⇒ we multiply each pixel by 2, so the new grey levels are in the range from 0 to 256, i.e.,

$$D_B = f(D_A) = 2D_A$$

`→Fig. 2.2`

Example 2: • in A, only the grey levels between D_{min} and D_{max} occur

`Figure: hist Dmin to Dmax`

⇒ first shift all values so that they start at 0:

$$D'_A = D_A - D_{min}$$

highest occurring value is now $D_{max} - D_{min}$ `Figure: hist 0 to (Dmax-Dmin)`

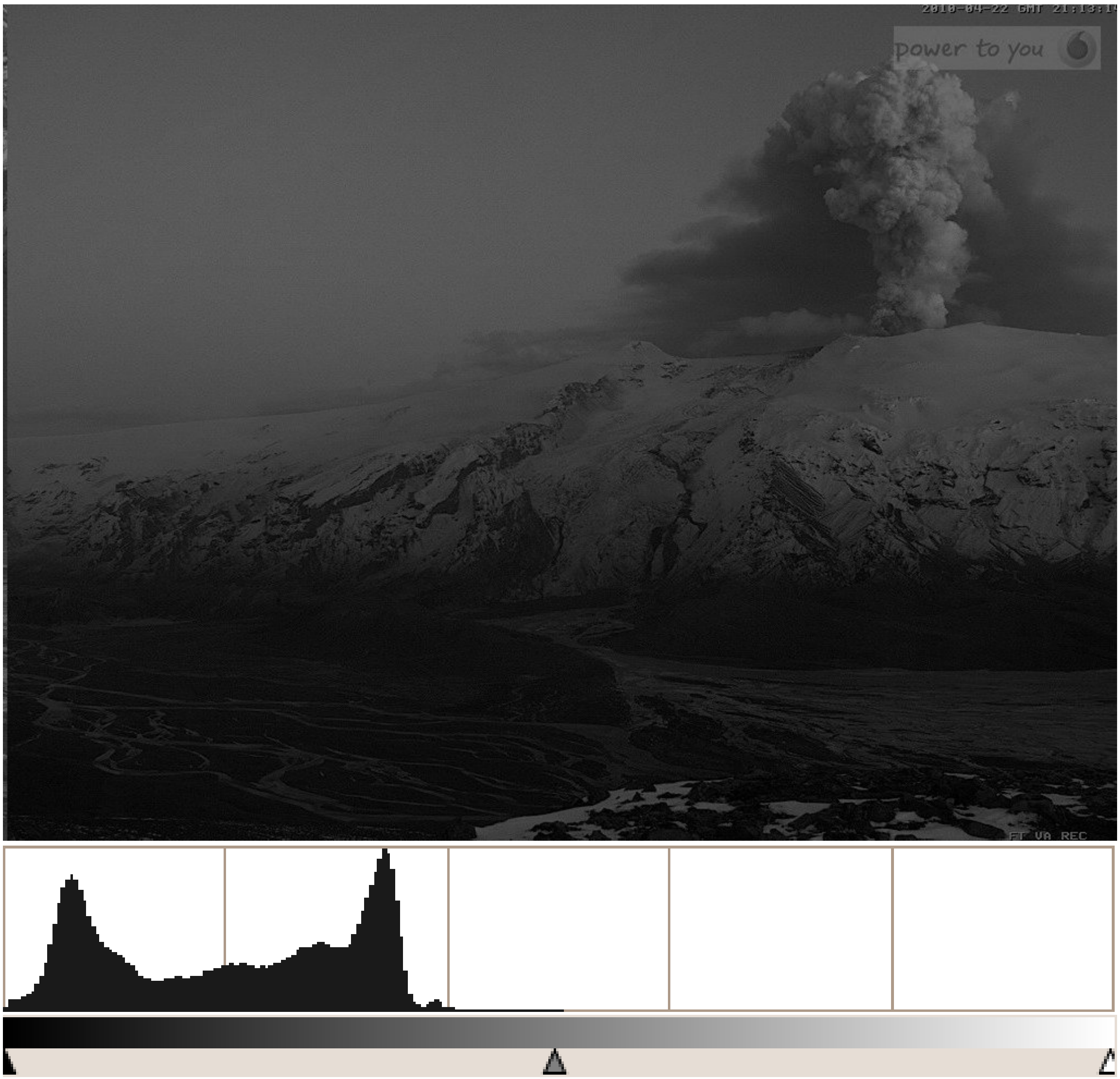

Fig. 2.1: *Image of the volcano Eyafjall in Iceland, April 2010 – very dark, only grey levels below 128 occur (source: web cam of Icelandic telecommunications provider Míla).*

⇒ then multiply by appropriate factor so that the highest values becomes, say, 255:

$$D_B = \frac{255}{D_{max} - D_{min}} D'_A$$

```
Figure:  hist 0 to 256
```

- This is a *linear contrast stretch*. Combined:

$$D_B = \frac{255}{D_{max} - D_{min}} (D_{max} - D_{min}) \tag{2.1}$$

- there are, of course, non-linear grey-level transformations.

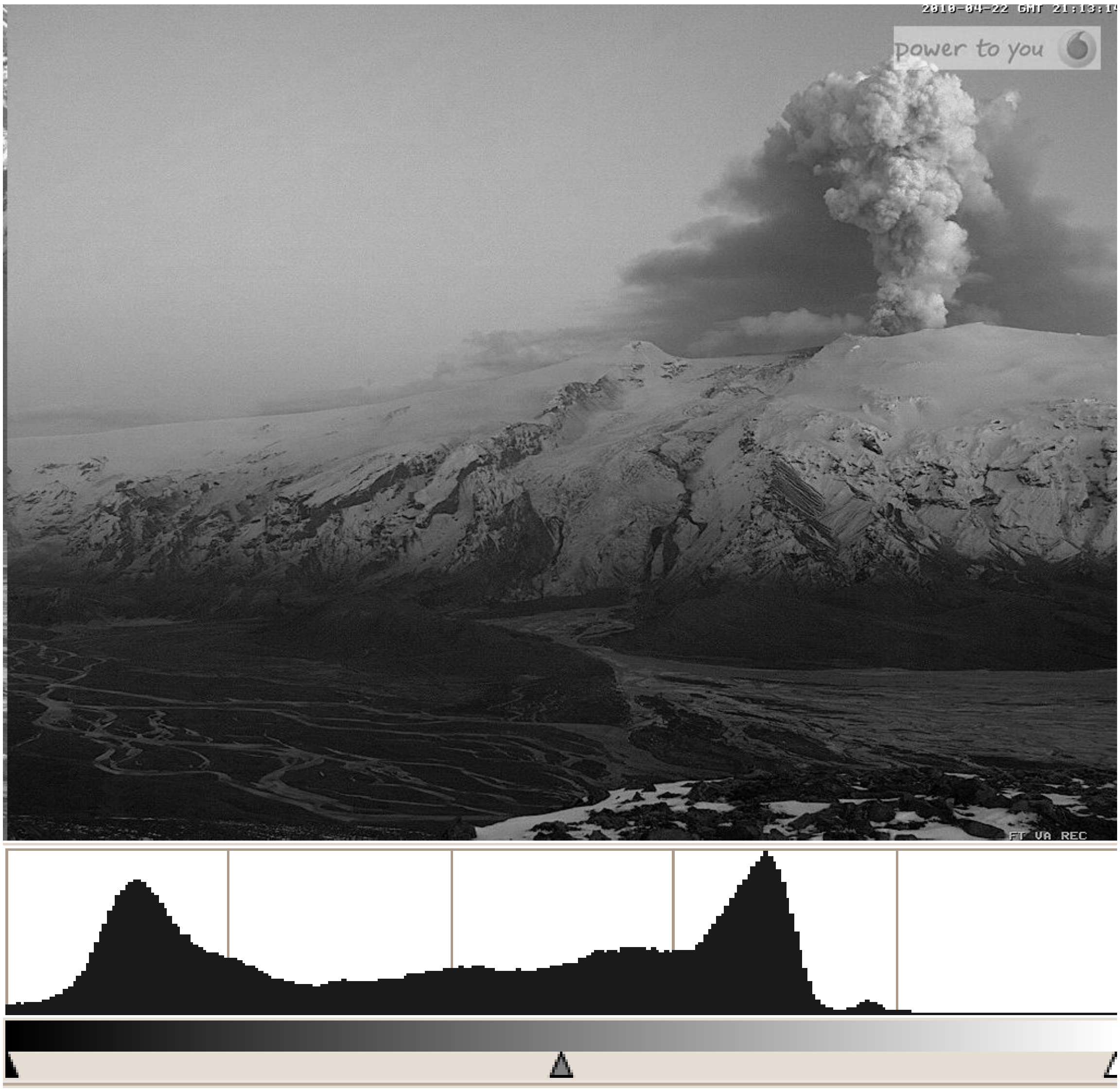

Fig. 2.2: *Same as previous image, but multiplied all pixel values by 2.*

- instead of writing down the function $f(D)$, its graph, called its *characteristic line*, is shown `→Fig. 2.3`
- Grey level transformations can also invert the grey levels (when char. line has negative slope)

2.2 Filters for Image Enhancement

- above: grey-level transformations for contrast enhancement (point operation)
- now: noise suppression or smoothing with local operations
- output pixel $B(m,n)$ depends on the pixel $A(m,n)$ and its neighbourhood N: `Figure: local operation`

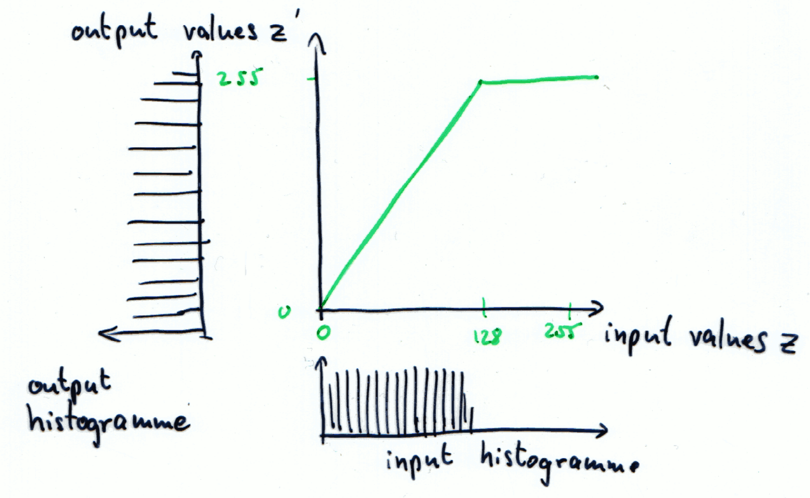

Fig. 2.3: *Characteristic line of a grey-level transformation.*

$$B(m,n) = f(\{A(m-m',n-n')|(m',n') \in N\} \tag{2.2}$$

where f is some function and the neighbourhood (the "moving window") can be defined as,e.g.,

$$\begin{aligned} N = & \{(-1,-1),(-1,0),(-1,1), \\ & (0,-1),(0,0),(0,1), \\ & (1,-1),(1,0),(1,1)\} \end{aligned} \tag{2.3}$$

(this is a 3 by 3 neighbourhood)

- local operations are often called filters

2.2.1 Linear Filters

- important group: linear filters

$$B(m,n) = \sum_{m'}\sum_{n'} H(m',n')A(m-m',n-n')) = H * A \tag{2.4}$$

this is the mathematical operation of a (discrete) convolution (symbol: $*$)

`Figure: application of convolution filter`

- H is a matrix of the size of the neighbourhood, and is called the
  - convolution kernel
  - filter kernel
  - point spread function

  - filter mask

- what Eq. (2.4) means (how two apply a linear filter):
  - place kernel H on the image A (centre of H at position (m,n))
  - multiply value of each element of H with the pixel value of the image A at that position
  - sum up everything and assign the result to new image B at the position (m,n) (where the centre of H is)
  - do this for all positions

- linearity:

$$(H+G)*A = H*A+G*A \tag{2.5}$$

$$H*(A+B) = H*A+H*B \tag{2.6}$$

$$H*(\alpha A) = \alpha(H*A) \tag{2.7}$$

  where α is some constant

- Important example: unweighted moving average, or unweighted mean:

$$H = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.8}$$

- Effect: smoothing, i.e. suppression of random noise

- but: smoothing also means blurring: small structures and edges (boundaries between areas of different grey levels) become less distinct!

- Another example: weighted mean:

$$H = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.9}$$

- Sum of all filter elements should be 1 (thus the normalisation factor) in order to keep the overall image brightness

  `Figure: example for effect of smoothing`

Notes:

1. problem when applying a filter at the edge of an image (border pixels): the filter kernel H would “stick out” over the edge of the image! 3 possible strategies:
   (a) do not apply filter near the edge when it would not entirely fit into the image (for a filter of size 5×5, this would be within 2 pixels from the border)
   (b) pad the image with zeroes (but this causes strange filter results near the border)
   (c) replicate the values of border pixels outside the image when necessary
2. square filter kernels with odd-numbered side length are convenient (well-defined centre pixel), but filter kernel need not be square

2.2.2 Non-Linear Filters

- Some simple, but non-linear filters can suppress noise without much blurring

Order-statistic filters: Median

- Most important non-linear filter: Median filter

$$B(m,n) = \text{median}(\{A(m-m',n-n')|(m',n') \in N\}) \tag{2.10}$$

  i.e. sort all pixels in the neighbourhood by value, take the middle one (the median)

- removes outliers
- preserves edges (but not corners or lines)
- size of window matters
- shape of window (cross, line) modifies effect on lines and corners

Other non-linear filters

- A filter for noise suppression without blurring:
  - compare pixel (X) with its neighbourhood (S_i) →Fig. 2.4

S_8	S_1	S_2
S_7	X	S_3
S_6	S_5	S_4

Fig. 2.4: *Pixel and its 8-neighborhood*