Liste der noch zu erledigenden Punkte

Kapitel 1

1.Overview

- „image society“ (webpages: 1995 text-based, 2005 image based, 2015 video based . . .)

  - data transfer rates $\uparrow$, compression rates $\uparrow$

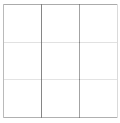
  critical shift: reading $\rightarrow$ watching

- „Photoshop“-ing (remove wrinkles, bumps, . . .)

- Images in medicine („medical image proscessing“), x-ray, CT, MRI, ultrasound, . . . („modalities“).
  different questions:

  1.) Layout!

  align bottom measurments $\overset{?}{\Rightarrow}$ image
  expl: tomography
  $\Rightarrow$ difficult mathematical problems

  2.) Image enhancements

  – denoising
    simple pixels/lines: „sandpaper“ interpolation so richtig?
    global noise: smoothing
  – grayscale
    histogramm balancing (spreading)
  – distortion
    makes straight lines (in real world) straight (in the images)
  – edge detection
    contour enhancement
  – segmentation
    detect and separate parts of the image
  – registration
    sequence of images of the same object $\Rightarrow$ Wort? , compare Skizze
    $\nearrow$ object following in a movie

<u>Our Focus:</u>

- mathematical models/methods/ideas
- (algorthms)
- ((implementation))

skipped: Very fast intro: Matlab and images

Kapitel 2

2.What is an image?

2.1 Discrete and continuous images

There are (at least) two different points of view:

Discrete / Digital

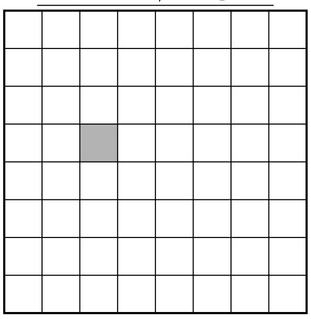


Abbildung 2.1: Discrete Image

Continous / Analogue

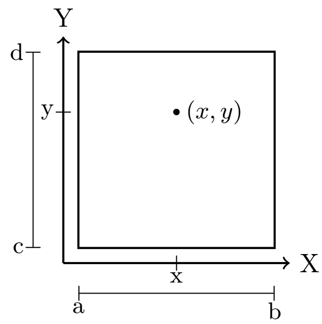


Abbildung 2.2: Continous Image

object:	matrix	function
tools:	linear algebra (SVD, . . .)	analysis (differentrage, integrate, . . .)
pros:	(finite storage) storage, complexity	freedom, tools, motions?P.4 (e.g. edge discontinuity)
cons:	limitations: zooming, rotations, . . .	storage (infinite amout of data)

arguably, one has:

- real life $\Rightarrow$ continuous „images“ (objects)
- digital camers $\Rightarrow$ discrete images

In general we will say:

Definition 2.1 ((mathematical) image)**.** A (mathematical) *image* is a function

$$u : \Omega \to F,$$

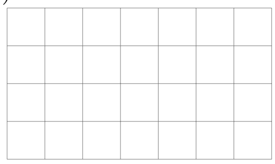
where: $\Omega \subset \mathbb{Z}^d$ (discrete) or $\Omega \subset \mathbb{R}^d$ (continuous) . . . *domain*
$d = 2$ (typical case 2D), $d = 3$ („3D image“ $=$ body or $\underbrace{\text{2D} + \text{time}}_{\text{movie}}$)
$d = 4$ (3D $+$ time)

F ... *range of colours*

$F = \mathbb{R}$ or $[0, \infty]$ or $[0, 1]$ or $\{0, \dots 255\}$, ... grayscale (light intensity)

$F \subset \mathbb{R}^3$ ... RGB image (colored)

$F = \{0, 1\}$ ... black/white



3 Layers
$\Rightarrow$ colored images:w

Matlab stuff

Large parts of the course: analytical approach (i.e. continuous domain Ω)
Since we want to differentirate, ... the image u.

Still: need to assume that also F ist continuous (not as $\{0, 1\}$, $\{0, 1, \dots, 255\}$ or $\mathbb{N}$) since otherwise the only differentiable (actually, the only continuous) functions $u : \Omega \to F$ are *constant* functions $\Leftrightarrow$ single-colour images

Also: We usually take F one-dimensional ($F \subset \mathbb{R}$). Think of it as either

- gray scaled image, or
- treating R,G & B layer separately

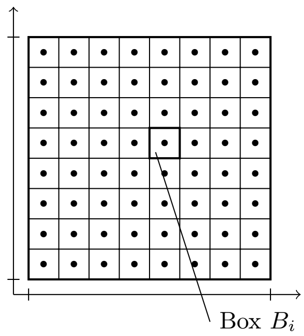
2.2 Switching between discrete and continuous images

continuous $\to$ discrete:

- divide the continuous image in small squared pieces (boxes) (superimpose grid)
- now: represent each box by *one* value
  - strategy 1: take function value $u(x_i)$ for x_i = midpoint of box B_i
  - strategy 2: use mean value

$$\frac{1}{|B_i|} \int_{B_i} u(x)dx$$

$\Omega =$



$\Rightarrow$ discrete image

strategy 1: simple (and quick) but problemativc ($u(x_i)$ might represent $u|_{B_i}$ badly; for $u \in L^p$, single point evaluation not even defined)

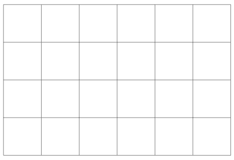
strategy 2: more complex but also more „democratic“ (actually closer to the way how CCD Sensors in digital camers work)

often the image value of the box B_i gets also digitized, i.e. fitted (by scaling & rounding) into range $\{0, 1, dots, 255\}$

discrete $\to$ continous

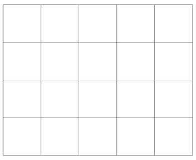
This is of course more tricky ...

- Again: each pixel of the discrete image corresponds to a „box“ of the continuous image (that is still to be constructed)
- Usually: pixel value $\mapsto$ function value at the *midpoint* of the box
- Question: How to get the other function values (in the box)?



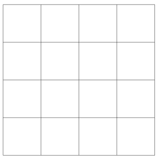
<u>idea 1:</u> just take the function value of the nearest midpoint („nearest neighbour interpolation“)

$$\text{For each } x \in B_i : u(x) := u(x_j) \quad \text{where } |x - x_j| = \min_k |x - x_k|$$



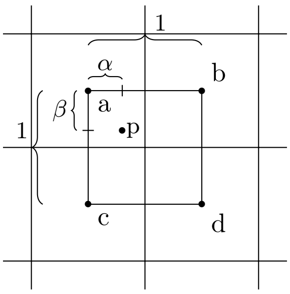
$\Rightarrow$ $u(x) = u(x_i)$ for all $x \in B_i$
$\Rightarrow$ each box is uni-color
$\Rightarrow$ the continuous image is essentially still discrete

<u>idea 2</u>: (bi-) lineare interpolation



Let $a, b, c, d \ldots$ function values at 4 surrounding adjacent midpoints ($\nearrow$ figure)
$\alpha, \beta, 1-\alpha, 1-\beta \ldots$ distance to dotted lines ($\nearrow$ figure, w.l.o.g, bob is 1×1)

interpolation (linear) on the dotted line between a and b:



$e := a + \alpha(b - a) = (1 - \alpha)a + \alpha b$
(1D - interpolation, convex combination)

Similarly: $\quad f = (1 - \alpha)c + \alpha d$

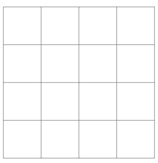
Then: The same 1D-interpolation between e and f

$$\begin{aligned}
\Rightarrow u(x) :&= (1-\beta) \cdot e + \beta \cdot f \\
&= (1-\beta)[(1-\alpha)a + \alpha b] + \beta[(1-\alpha)c + \alpha d] \\
&= \underbrace{(1-\alpha)(1-\beta)}\, a + \underbrace{\alpha(1-\beta)}\, b + \underbrace{(1-\alpha)\beta}\, c + \underbrace{\alpha\beta}\, d
\end{aligned}$$

$$\in [0, 1] \wedge \sum = 1$$

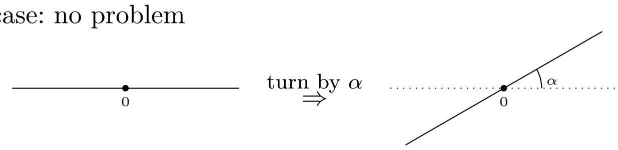
$\Rightarrow$ <u>convex combination</u> of the function values a, b, c, d at the the surrounding 4 midpoints (on which points is the nearest instead of taking just a, b, c or d - depending)
$\Rightarrow$ 2D linear interpolation, *bi-linear interpolation* (can be interpreted as spline interpolation with bilinear basis splines).

Beispiel 2.2. Rotate image



by angle $\phi \neq k \cdot \frac{\pi}{2}$

- continuous image case: no problem



$$x = D_\varphi\, y \qquad x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix},\ y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix},\ D_\varphi = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix}$$

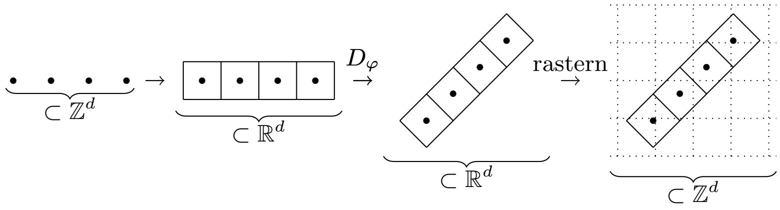
2D rotation matrix

$$y = D_\varphi^{-1}\, x = D_{-\varphi}\, x$$

$$\Rightarrow v(x) := u(y) = u(D_{-\varphi}\, x) \quad \forall x \in \text{domain of the rotated image}$$

- discrete image case: problem !

  For $x \in$ domain of notated image, in general $D_{-\varphi}\, x \notin$ domain of original image[1]
  Way out: $v(x) :=$ *interpolation* between the $u(\cdot)$ of the 4 surrounding pixels of $D_{-\varphi}$



Something to think about:
What happens in the limit (?) if we, starting with an image (discrete or continuous), repeatedly switch between discrete and continuous, non-stop ...?
Does the answer depend on the way of switching ? (continuous $\to$ discrete: midpoint or average, discrete $\to$ continuous: nearest neighbour or bilinear?)

[1] it's not an integer

Kapitel 3

3.Histogramm and first applicatsion

3.1 The histogramm

Definition 3.1 (histogram)**.** Let $\Omega \subset \mathbb{Z}^d$, $F \subset \mathbb{R}$ discrete and $u : \Omega \to F$ a discrete discrete image. The function

$$H_u : F \to \mathbb{N}_0 \; (:= \mathbb{N} \cup \{0\})$$

with

$$H_u(k) := \# \{x \in \Omega : u(x) = k\}, \quad k \in F$$

is called *histogramm* of the image u.

$H_u(k)$ counts how often colour k appears in u.

$$\sum_{k \in F} H_u(k) = |\Omega| = \text{number of pixels in the whole image}$$

or

$$\frac{H_u(k)}{|\Omega|} = \text{relative frequence of colour } k \text{ in image } u \quad \text{(relative Häufigkeit)}$$

Beispiel 3.2. $u =$

has $H_n =$

If u ist a continous image, H_u can be understood as measure (generalized function)[1].
Another way to writ this:

$$H_u(k) = \sum_{x \in \Omega} \delta_{u(x)}(k), \; k \in F \qquad H_u(k) = \int_\Omega \delta_{u(x)}(k) dx, \; k \in F$$

hier fehlt noch das Kronecker underarrow

Matlab-Code

3.2 Application: contrast enhancement

If the image only uses a small part of the available colour/grayscale „palette“ F, then its contrast can be improved by „spreading“ the histogramm over all of F.
Simple idea:

[1] density of a probability distribution

$$v = \varphi \circ u$$
$$v(k) = \varphi(u(k))$$

The above simple idea („contrast stretching“) corresponds to

$$\begin{aligned} \varphi : k_{\min} &\mapsto 0 \\ k_{\max} &\mapsto N \\ &\text{and linear in between} \end{aligned}$$

$$\text{i.e} \qquad \varphi(k) = \left[\frac{k - k_{\min}}{k_{\max} - k_{\min}} \right]$$

Where $[\quad \cdot \quad]$ means . . . rounding to the nearest integer (assumuning that $F = \{0, 1, \ldots, N\}$.

A bit more sophisticated:

$$\begin{aligned} \varphi : \ (k_{\min} &\mapsto 0) \\ k_{\max} &\mapsto N \\ &\text{and } \textbf{non} \text{ linear in between} \end{aligned}$$

such that colour ranges that occur more frequently in u can occupy a larger range of colours in u. ($\Rightarrow$ visibility $\uparrow$)

Example histogramm:

spread out according to frequency of occurence:

$\Rightarrow$ „density“ is equalized over $F = \{0, \ldots, N\}$

Ideal would be:

Layout S.12 u

Note: The new colours (i.e the location of the bars in the histogramm of u) only depend on the frequencies / *height* of the bars in H_u but not on the colours/location of the bars in H_u

Finally: The formula

$$\varphi(k) = \left[\frac{N}{|\Omega|} \sum_{l=0}^{k} H_u(l) \right]$$

This process is called „histogramm equalization“

Exercise ?!

3.3 Another application: conversion to b/w