









Liste der noch zu erledigenden Punkte

 Layout!	2
 so richtig?	2
 Wort?	2
 Skizze	2
 skipped: Very fast intro: Matlab and images	2
 motions?P.4	3
 Matlab stuff	4
 basis	5

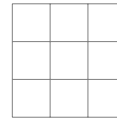
Kapitel 1

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 processing“), x-ray, CT, MRI, ultrasound, ... („modalities“).
different questions:

1.) Layout!

align bottom measurements $\stackrel{?}{\Rightarrow}$ image
 expl: tomography
 \Rightarrow difficult mathematical problems



2.) Image enhancements

- denoising
 - simple pixels/lines: „sandpaper“ interpolation
 - 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

so richtig?

Our Focus:

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

skipped: Very fast intro: Matlab and images

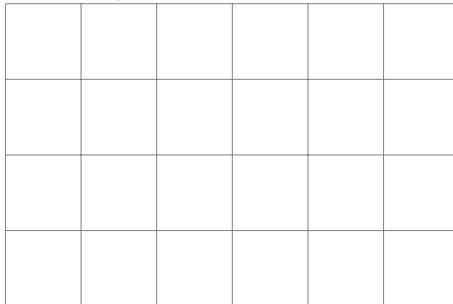
Kapitel 2

2. What is an image?

2.1 Discrete and continuous images

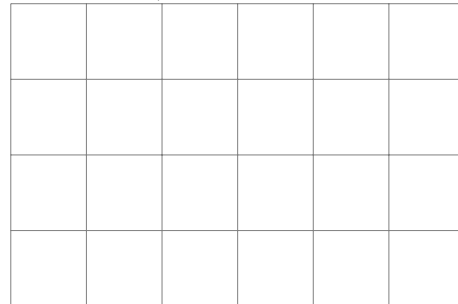
There are (at least) two different points of view:

• discrete/digital



object: matrix
tools: linear algebra (SVD, ...)
pros: (finite storage) storage, complexity
cons: limitations: zooming, rotations, ...

• continuous/analogue



function
 analysis (differentiation, integrate, ...)
 freedom, tools, **motions?P.4**
 (e.g. edge discontinuity)
 storage (infinite amount of data)

arguably, one has:

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

In general we will say:

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

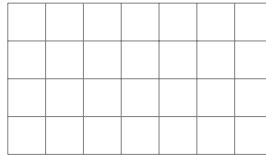
$$u : \Omega \rightarrow F,$$

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 $\underbrace{2D + time}_{\text{movie}}$)
 $d = 4$ (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 differentiate, ... the image u .

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

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

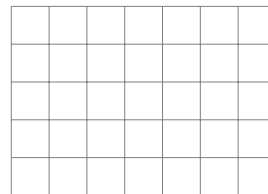
- gray scaled image, or
- treating R,G & B layer separately

2.2 Switching between discrete and continuous images

continuous \rightarrow 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 $x_i = \text{midpoint of box } B_i$
 - strategy 2: use mean value

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



\Rightarrow discrete image

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

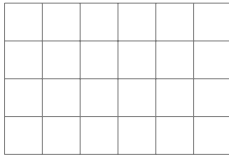
strategy 2: more komplex but also more „democratic“ (actually closer to the way how CCD Sensors in digital camers work)

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 \rightarrow continuous

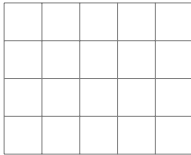
This is of course more tricky ...

- 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)?



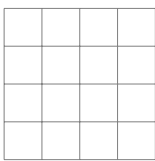
idea 1: just take the function value of the nearest midpoint („nearest neighbour interpolation“)

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



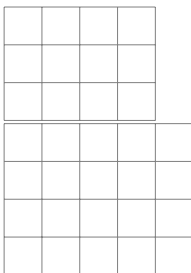
$\Rightarrow u(x) = u(x_i)$ for all $x \in B_i$
 \Rightarrow each box is uni-color
 \Rightarrow the continuous image is essentially still discrete

idea 2: (bi-) linear interpolation



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

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

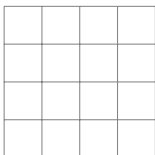


$$e := a + \alpha(b - a) = (1 - \alpha)a + \alpha b$$

(1D - interpolation, convex combination)

$$\text{similarly: } f = (1 - \beta)c + \beta d$$

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

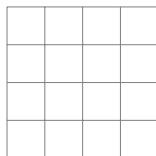


$$\begin{aligned}
 \Rightarrow u(x) &:= (1 - \beta) \cdot e + \beta \cdot f \\
 &= (1 - \beta)[(1 - \alpha)a + \alpha b] + \beta[(1 - \beta)c + \beta d] \\
 &= \underbrace{(1 - \alpha)(1 - \beta)a}_{\in [0, 1]} + \underbrace{\alpha(1 - \beta)b}_{\wedge \Sigma = 1} + \underbrace{(1 - \alpha)\beta c}_{\in [0, 1]} + \underbrace{\alpha\beta d}_{\wedge \Sigma = 1}
 \end{aligned}$$

\Rightarrow convex combination 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)

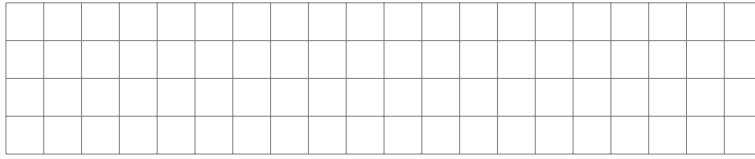
\Rightarrow 2D linear interpolation, *bi-linear interpolation* (can be interpreted as spline interpolation with bilinear basis splines).

Beispiel 2.2. Rotate image



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

- continuous image case: no problem



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

2D rotation matrix

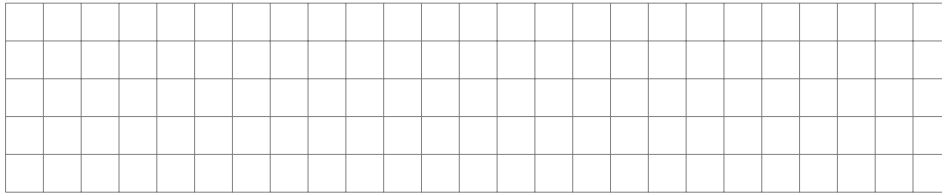
$$y = D_\varphi^{-1} x = D_{-\varphi} x$$

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

- discrete image case: problem !

For $x \in \text{domain of notated image}$, in general $D_{-\varphi} x \notin \text{domain of original image}$ ¹

Way out: $v(x) := \text{interpolation}$ between the $u(\cdot)$ of the 4 surrounding pixels of $D_{-\varphi} x$



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 \rightarrow discrete: midpoint or average, discrete \rightarrow continuous: nearest neighbour or bilinear?)

¹it's not an integer

Kapitel 3

3. Histogramm and first applications

3.1 The histogram

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

$$H_u : F \rightarrow \mathbb{N}_0 \quad (:= \mathbb{N} \cup \{0\})$$

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

$$\frac{H_u(k)}{|\Omega|} = \begin{array}{l} \text{relative frequency of colour } k \text{ in image } u \\ \text{(relative H\u00e4ufigkeit)} \end{array}$$

Beispiel 3.2. $u =$

 has $H_n =$
