UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

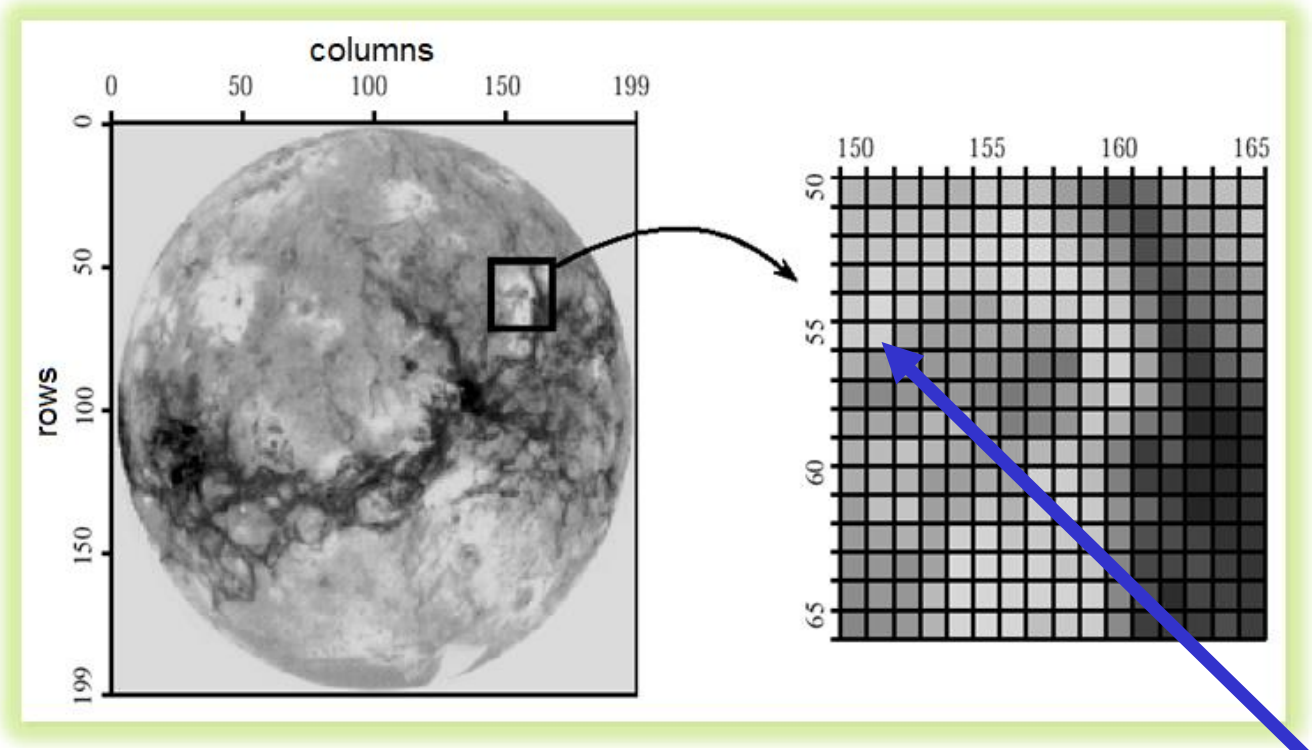
Image Processing

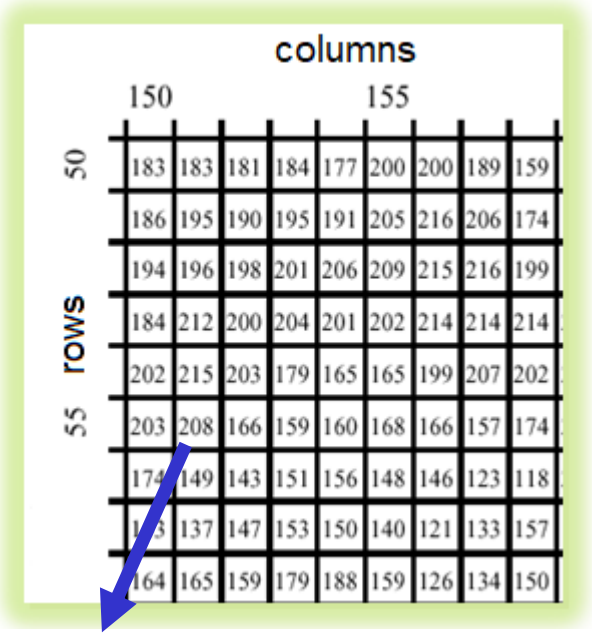
Introduction and Basics

DI(FH) Dr. Gerald Adam Zwettler, MSc, gerald.zwettler@fh-hagenberg.at
Lector of Computer Graphics, Image Processing and Computer Vision

Digital Images

- digital images represented as 2D mask of scalar measured values
  - access to data via index function
  - representation of data with grey or colour values (true or pseudo colours)





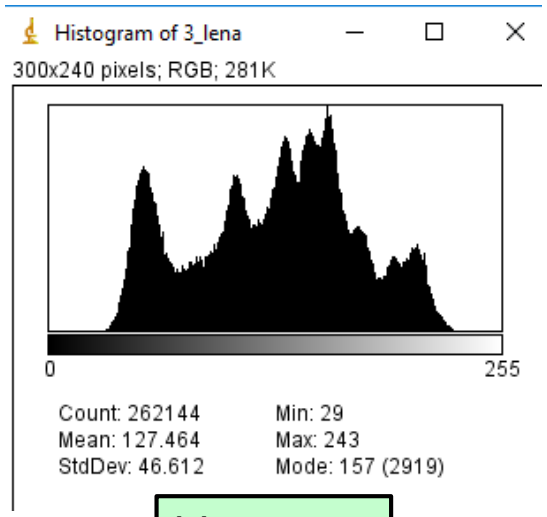
  - access to scalar values via coordinate notation, e.g. $I(151,55) = 208$ at "***pixel***" $(151,55)$ i.e. *picture element*

Digital Images – Mathematical Definition

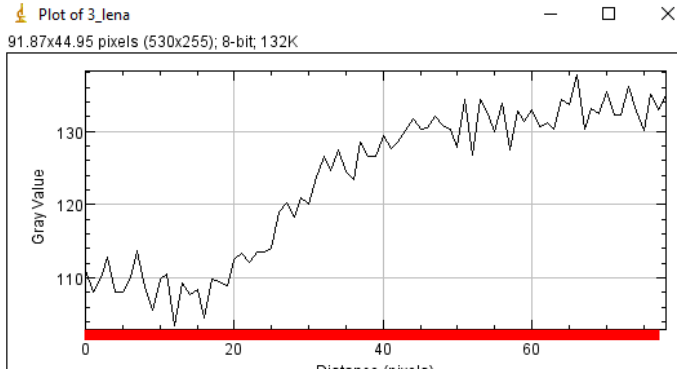
- dual nature of images: *function* and *set* of discrete coordinates
  - a 2D image ***I*** is generally a mapping $f: \mathbb{N}^2 \rightarrow \mathbb{N}$ and thus represents a function $I(x, y) \in \mathbb{N}$ with $x, y \in \mathbb{N}^{\geq 0}$ thereby projecting from coordinate space $[0, \dots, N_x] \times [0, \dots, N_y] \mapsto \{0,1, \dots, g-1\}$ and $g \in \mathbb{N}^{\geq 0}$ as maximum scalar range. The maximum indices per dimension N_x and N_y are denoted as $width(I)$ and $height(I)$ respectively.
  - in case of negative indices/scalar values, offsets are applicable in general
  - the coordinates of an image ***I*** are represented by a set ***C*** of two-dimensional tuples as $\{c_i\}$ with $c_i = (c_{i,x}, c_{i,y})$.The image ***I*** thereby can only be evaluated at discrete positions $c_i \subseteq C$ with $I(c_i) = I(c_{i,x}, c_{i,y}) \mapsto \{0,1, \dots, g-1\}$
  - a sub image ***R***, i.e. ROI (region of interest) or segmented pixel mask, thus is represented by a subset of coordinates $R \subseteq C$.
  - common image resolutions are 1280×720 pixels for web cams, 2960×1440 for smart phones (Samsung Galaxy S8) and rather with iso-aspect-ratio in medicine (256×256 , 512×512 , 1024×1024).
  - relevant scalar range e.g. [0;255] for 8-bit unsigned char

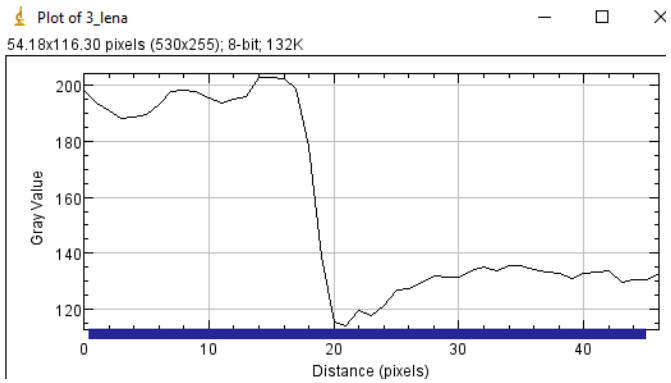
Histogram

- global distribution of the scalar values of an image A generally plotted as histogram
  - thereby, frequency per bin (i.e. discrete scalar value or scalar range) is plotted as bar chart
  - histograms allow analysis of contrast (high, low, dark, light,...) and the intensity deviation of semantic objects in the image data
- local gradients and edge magnitude to be analysed utilizing line-profiles
  - level of background noise and object-related contrast clearly visible



histogram

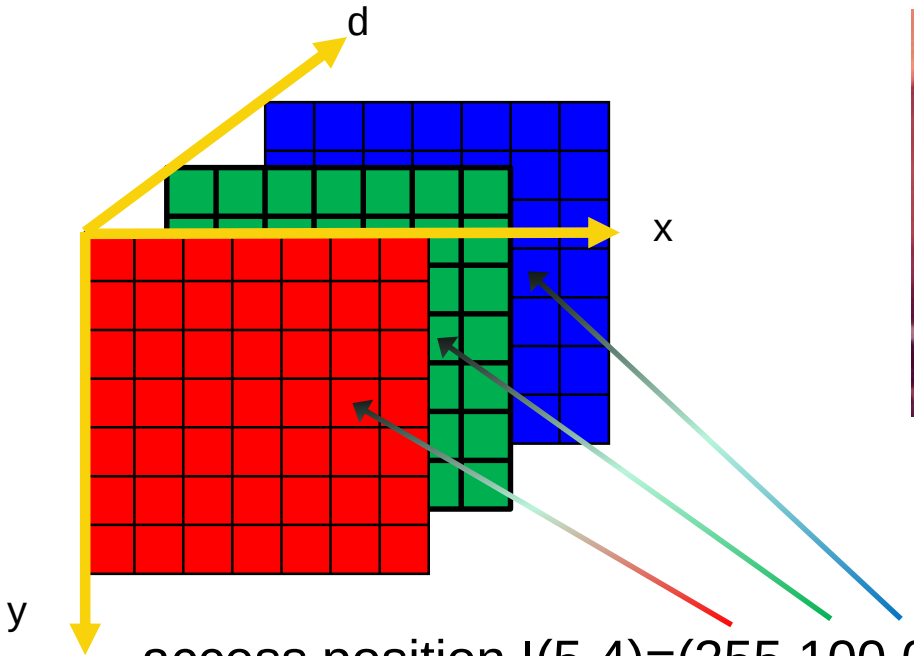




line profile in background and edge area

Colour Images

- for most relevant image formats, colour is represented by scalar values per separate colour channel, mainly as RGB + optional alpha (opacity)
  - alternatively, the colour channels might be encoded in a single scalar value, e.g. 4 byte with a byte resulting from the colour values and the alpha channel necessitating for shift operations to access
  - in case of separate colour channels, the coordinate access is extended by the colour index or returning a vector of scalar values



Lena (or Lenna)

red

green

blue

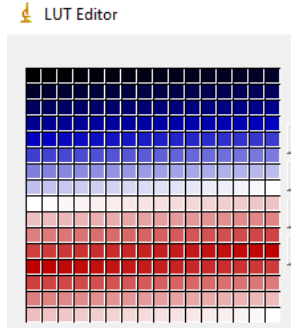
  - access position I(5,4)=(255,100,0) or access specific colour channel I(5,4,1)=100

Conversion From Colour to Grey Values

- in case of converting from colour information to grey values, luminosity (*Helligkeit*) is conserved, while chromaticity (*Farbwert*) is lost.
  - *average* method as the most intuitive approach
    - $g = \frac{(R+G+B)}{3}$
  - *lightness* method balancing statistical mean shift tendency
    - $g = \frac{\left(max(R,G,B)+min(R,G,B)\right)}{2}$
  - *luminosity* method inspired by human visual perception
    - e.g. $g = 0.21R + 0.72G + 0.07B$

Colour via Lookup Table

- scalar values, e.g. from grey-value images, can be assigned a colour via indexed lookup table
  - thereby the (single channel) scalar representation remains unchanged, while the colouring is indirectly achieved vis LUTs



LUT Union Jack – all 16x16==256 possible scalar values get assigned a specific colour

LUT grey

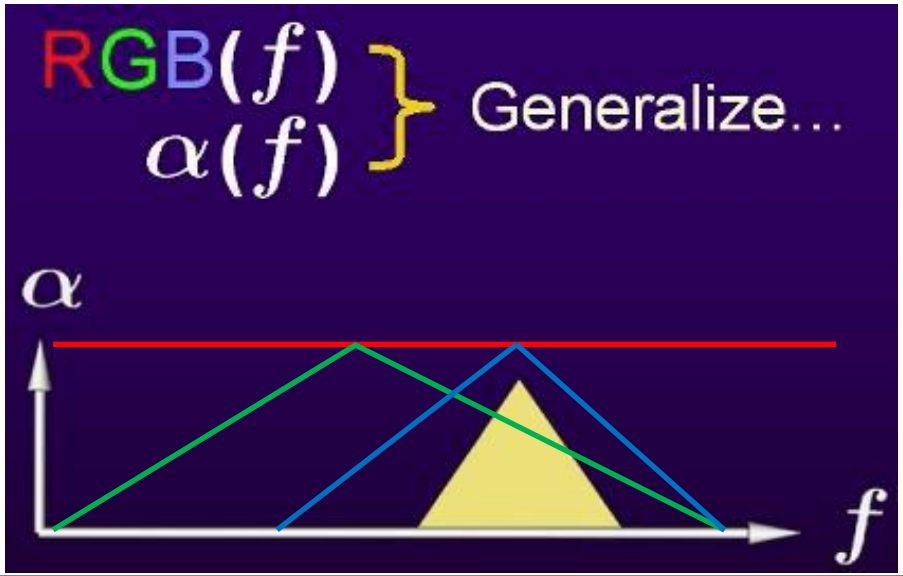
LUT Fire

LUT Glasbey

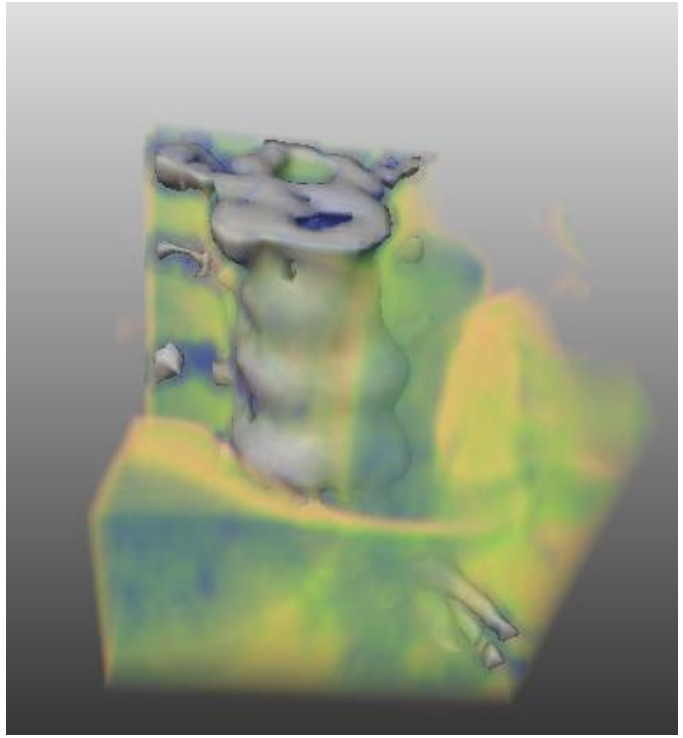
LUT Phase

Colour via Lookup Table

- assigning colours for full scalar range necessitates a lot of user interaction
  - thus, colour editors interpolate the colours based on a couple of explicitly defined control points
  - in case of 3D data, opacity linked to the amplitude as additional input parameter is required to get insight into the 3D volume



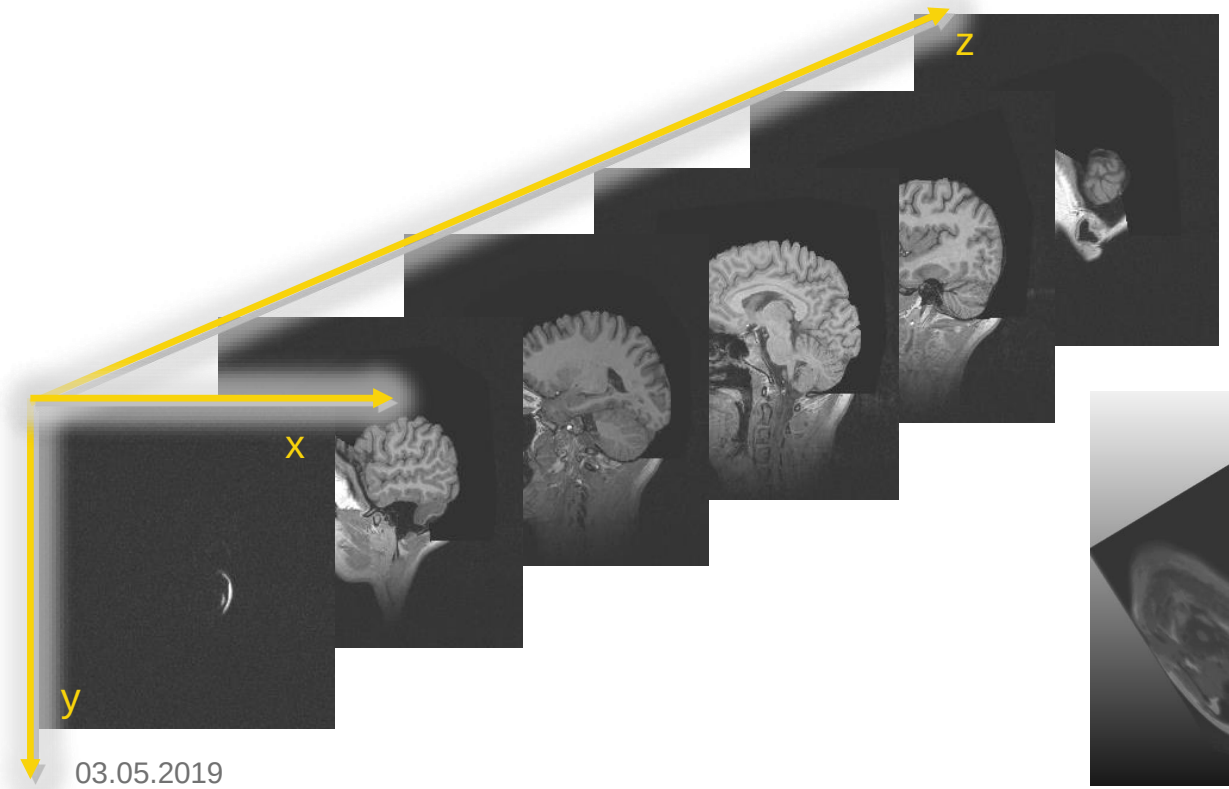
with only 4 explicit scalar colours assigned,
the full range is complemented utilizing interpolation

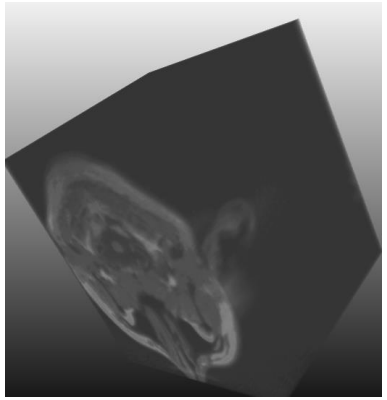


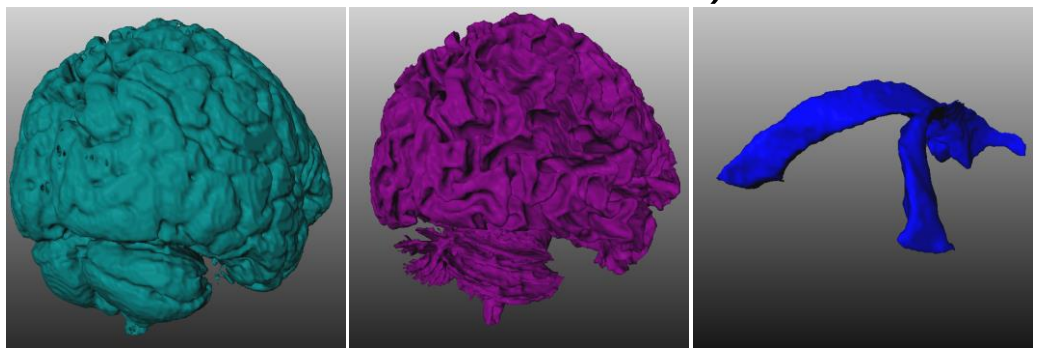
based on proper LUTs pseudo segmentations of 3D volumes become feasible

Three-dimensional Image Data

- in case of 3D image data (e.g. medical/industrial tomography from MR or CT, 3D scans, ultrasonic/laser scans,...), the volume is interpreted as stack of 2D images with “z-axis” as additional dimensionality and values present at local “*voxels*” (voxel elements).
- in case of animated time-series, 2D+t or 3D+t data, often denoted as 4D, is given
  - visualization techniques required to get sight into the dataset
  - volume as *tensor* of rank 3 (scalar with rank 0, vector with rank 1,...)







3D volume as solid block – necessitating segmentations/visualizations

Image Properties

- most image data formats comprise header information with meta data and the particular image content in a sequential data block
- the following image properties are relevant to unambiguously interpret binary content as image
  - **length** of the **header** (for most image formats either fixed size or at begin of the header)
  - **scalar data type** (int, char) with *signed/unsigned* and *byte order* besides optional offset for negative values
  - **size** per dimension (x,y,z)
  - **spacing** (in case of available real-world scale information, “Abbildungstreu”) or **pixel resolution** in ppi.
  - **origin** and **orientation** (left- or right-handed coordinate system)

Raw Image Data

- in case of loading an image in RAW format (plain byte sequence), the correct header information is inevitable for image interpretation
  - thus, pixel type and image size per dimension need to be correct. In case of pixel datatype >1 byte, the byte order is relevant too
  - for colour images, the interweaving of the colour channels needs to be considered too

12 bytes

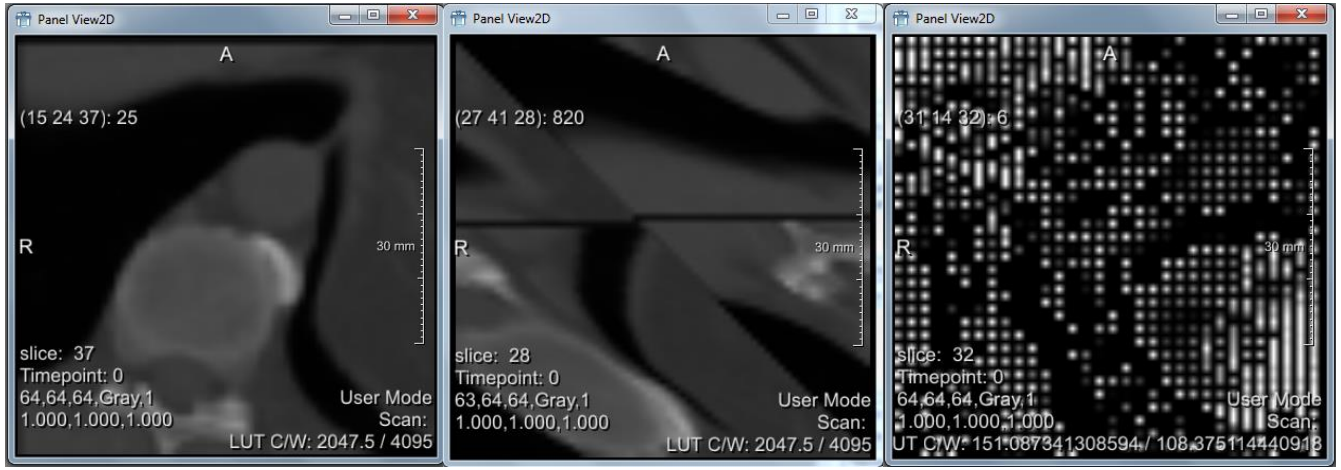
0	10	0	10	0	10	0	10	0	10	0	10

0	10	0	10
0	10	0	10
0	10	0	10

image reconstructed with 4x3 image size

0	10	0
10	0	10
0	10	0
10	0	10

image reconstructed with 3x4 image size



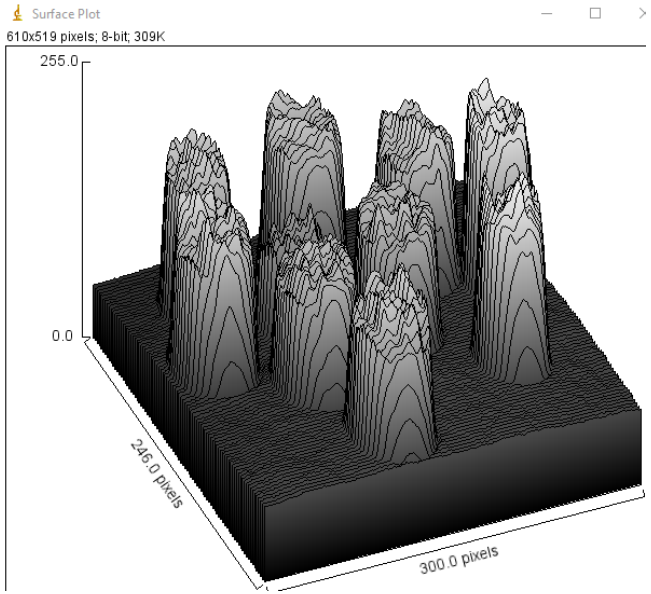
64x64x64, 2bytes

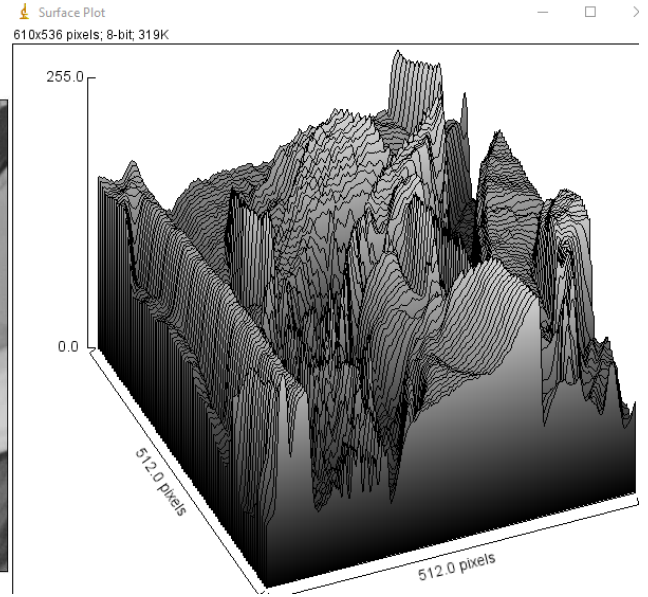
63x64x64, 2bytes

64x64x64, **1byte**

Images as Topography

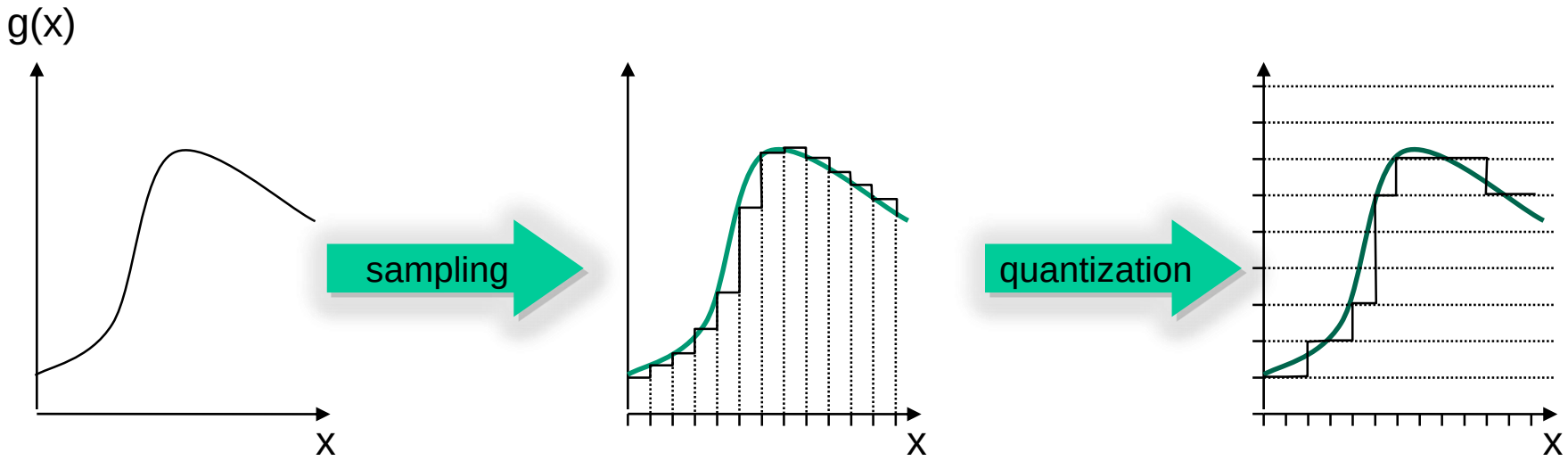
- any 2D image can be interpreted as 3D topography too
  - thereby, the scalar value per pixel is interpreted as height information.
  - thus, a topography with rises (high scalar values) and sinks (low scalar values) is manifested
  - in case of noise, the topography is rugged, while in case of homogeneous regions, the local area is smooth and horizontally aligned
  - aspects like noise level and edge/gradient magnitude to be observed in the topographic data





Images as Discrete Signals

- images as result of discretization/sampling, e.g. utilizing an A/D converter
  - *given*: continuous function or analogue signal
  - *wanted*: discrete function or digital signal
- sampling (*Abtastung*)
  - an analogues signal is evaluated at discrete positions (time stamps) only
- quantization
  - the values measured at the discrete positions are generally restricted to integer range



Signal Theory – Images as Waves

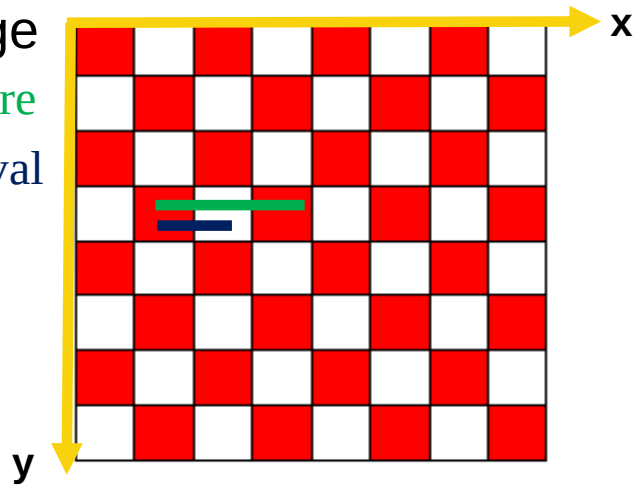
- wavelength $\boldsymbol{\lambda}$ – size of a (periodic) structure
  - for image data, “pixel” and “voxel” as unit for the wavelength
- spatial frequency $\boldsymbol{f}$ – number of repetitions of a (periodic) structure per length
  - $f = \frac{1}{\lambda}$
- the smallest structure to be represented in images shows a wavelength $\lambda_{min} = 2$ pixel.
  - thus, $f_{max} = \frac{1}{\lambda_{min}} = \frac{1}{2}$ per pixel
- the largest structure to be represented in images shows full size w.r.t. the particular dimension, e.g. 256 for images with border length $w = 256$
  - $f_{min} = \frac{1}{\lambda_{max}} = \frac{1}{256}$ per pixel

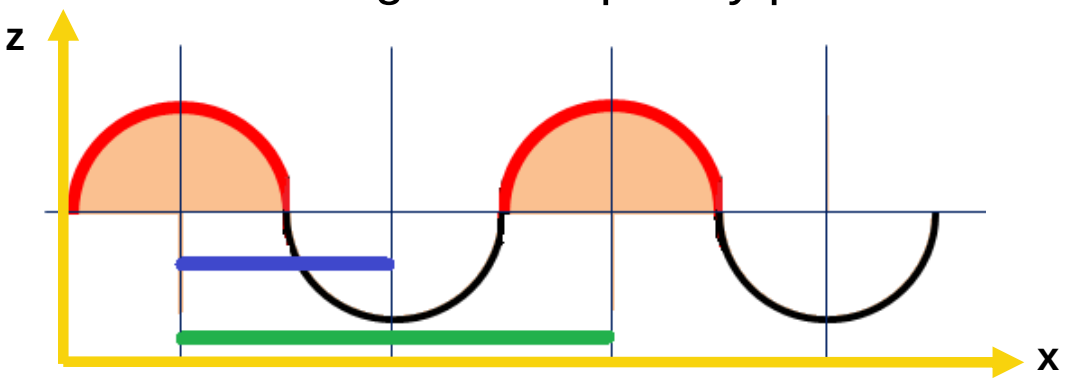
Signal Theory – Images as Waves cont'd

- the number of repetitions of a periodic structure is denoted as repetency ***u*** (wavenumber, "Wellenzahlindex")
  - in case of images, ***u*** is favoured over frequency ***f***, as all values ≥ 0
  - the maximum wavenumber ("Grenzwellenzahl") is denoted as $\boldsymbol{u_{max}}$
  - for image of size 512×512, the wavenumber is limited between $u_{max} = 256$ and $u_{min} = 1$
- *Nyquist* sampling theorem
  - $f_{sampling} \geq 2 \cdot f_{max}$
  - sampling frequency thereby must be chosen at least twice the highest frequency present in the particular image

smallest period/structure

sampling interval

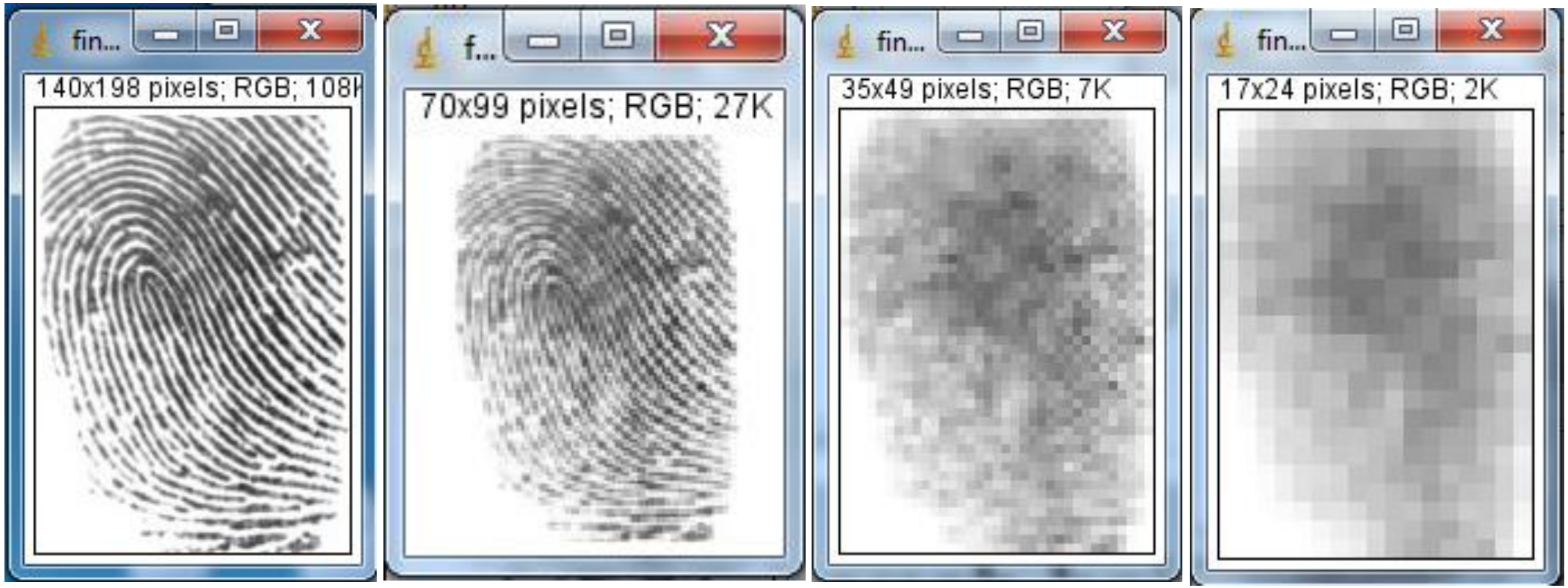




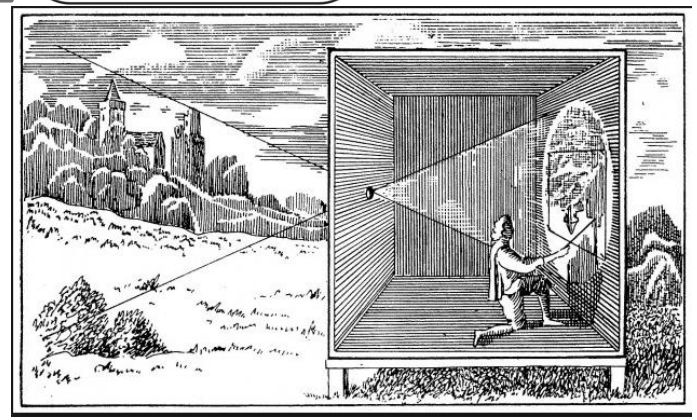
topographic cut through fixed y-value

Signal Theory – Images as Waves cont'd

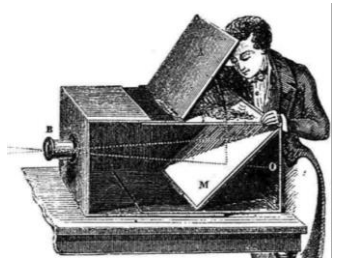
- in case of reducing the image size, the Nyquist theorem needs to be considered w.r.t. very small and thin structures
  - in the following example, the thin black lines of a fingerprint merge as the image size is scaled down

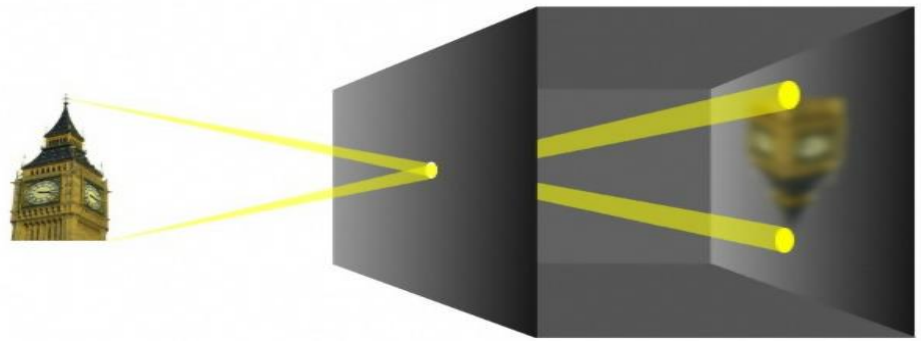


Imaging Systems



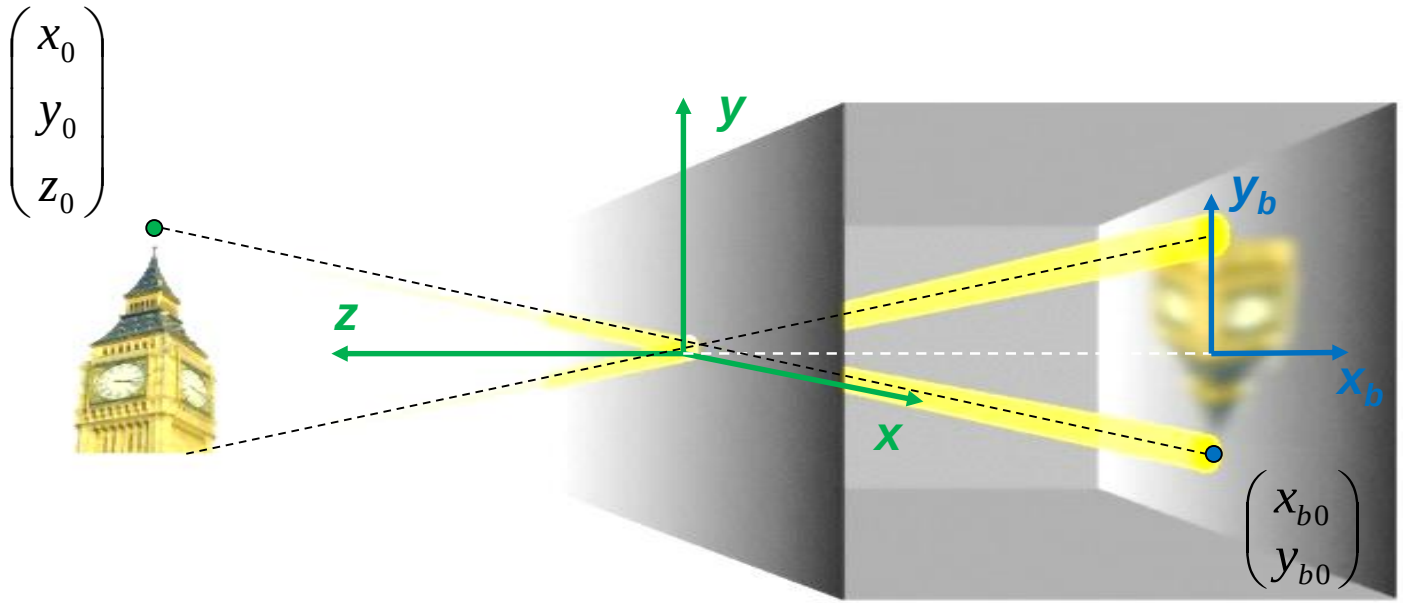
- The pinhole camera – *camera obscura*
  - most basic imaging system, known since ancient times
  - light shines onto the back wall of a dark box or cabin/room.
  - the object is reproduced (pictured) on the back image plane
  - size of the pinhole balances between sharpness and brightness
    - each point of the world (like top of *“Elisabeth Tower”* including comprising *“Big Ben”*) of minimal size contributes to a circular area of the image plane depending on size of the pinhole
    - thus, as several color rays of real world contribute for each point on the image plane, blurring is introduced
  - resulting images are inverted, i.e. mirrored top-down and left-right
  - ratio of optical axis vs. distance from pinhole to the image plane determines if the object is pictured larger or smaller compared to true size.



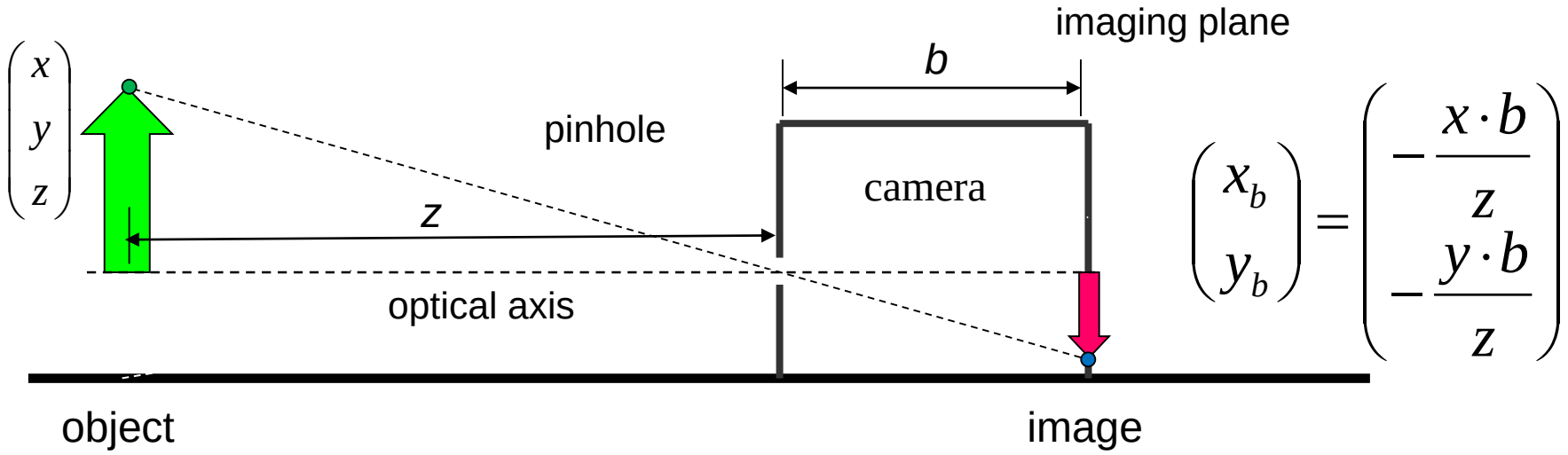


Imaging Systems

- *camera obscura cont'd*
  - Coordinate Systems
    - **camera coordinates x, y, z:** fixed to the camera, 3D, orthonormal and right-handed
    - **image coordinates x_b, y_b:** linked to the image plane, 2D, reflect projection geometry 3D➔2D



- *camera obscura – projection geometry of the imaging system*
  - negative sign, i.e. mirroring/inversion around the optical axis
  - in case $z < b$, thus distance to the object smaller than to the image
    - 2D image of the object is enlarged
  - in case $z > b$, thus distance to the object larger than to the image
    - 2D image of the object is scaled down



UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

Image Processing

Intensity Transformation and Image Operations

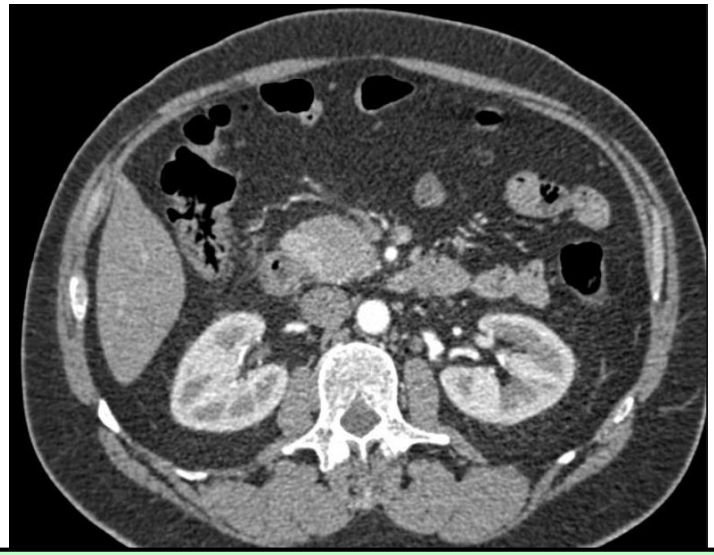
DI(FH) Dr. Gerald Adam Zwettler, MSc, gerald.zwettler@fh-hagenberg.at
Lector of Computer Graphics, Image Processing and Computer Vision

Intensity Transformations

- motivation:
  - images might lack visual contrast
  - marginal differences in scalar values can be interpreted by computers, but human eyes only can discriminate a discrete amount of grey values and colors.
  - thus, scalar range is adapted as a visualization technique (no further information value added!)

low-contrast gray value image – higher scalar values underrepresented

low-contrast color image



high-contrast gray value image with 12bit, 4096 values – fine details (lesions, vessels) not visible for human eye

Intensity Transformations

- intensities of output image ***B*** are calculated by transformation function $b_i = T(a_i)$
  - often denoted as scalar transformations
  - the mapping of the pixel intensities is thereby adapted in a static way
  - in contrast to convolution filtering, no weighting or local neighbourhood aspects are taken into consideration
- general description:
  - each possible or present intensity a_i of input image ***A*** is transformed to a different intensity b_i to build up the transformed result image ***B***.
  - the resulting intensities b_i are calculated exclusively based on a_i and/or histogram of input image **A**.
- mathematical formulation:

$$A = \{a_i | i = 1..N, a_i \geq a_{\min}, a_i \leq a_{\max}\}, B = \{b_i | i = 1..N, b_i \geq b_{\min}, b_i \leq b_{\max}\}$$

$$f : A \rightarrow B$$

$$b_i = T(a_i)$$

Intensity Transformations cont'd

- example: contrast/intensity inversion as a very simple intensity transformation
  - transformation function thereby represents a monotonic increasing/dropping or horizontal curve
  - not all values b_i must be part of the result set and several different intensities a_i, a_j might map to the same result intensity b_k

0	0	4
2	1	5
3	1	4

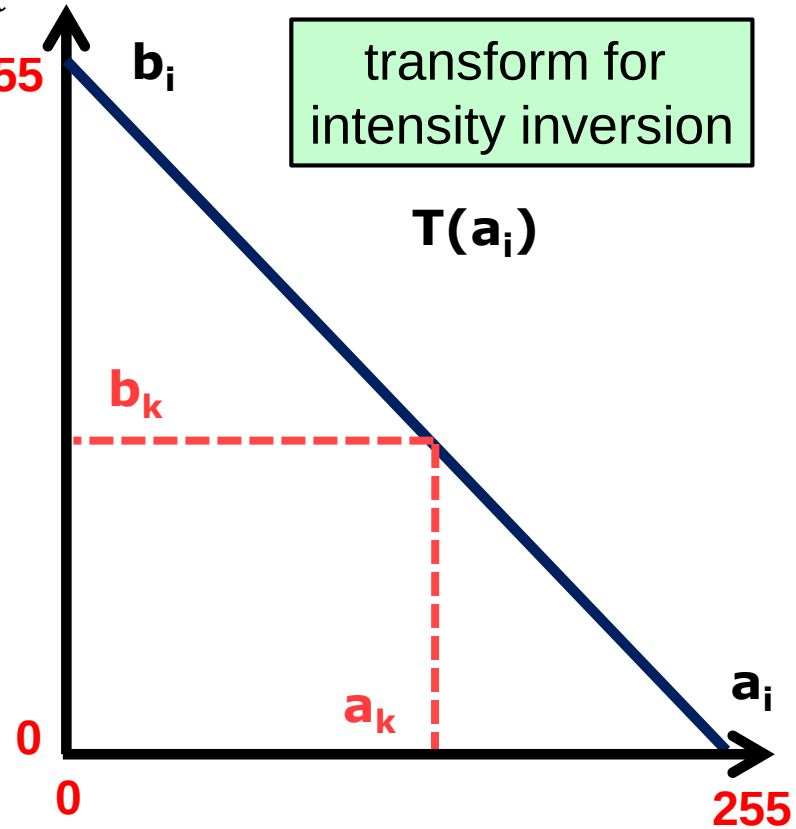
A

T(A)

5	5	1
3	4	0
2	4	1

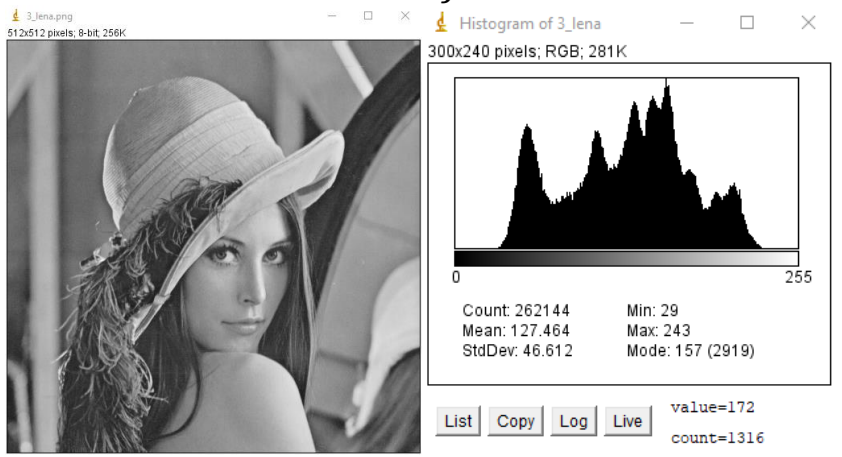
B

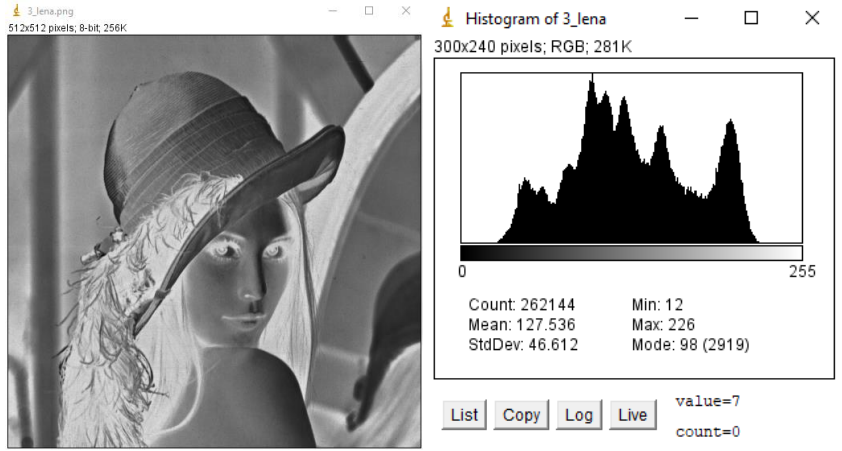
$$T(a_i) := a_{max} - a_i$$



transform for intensity inversion

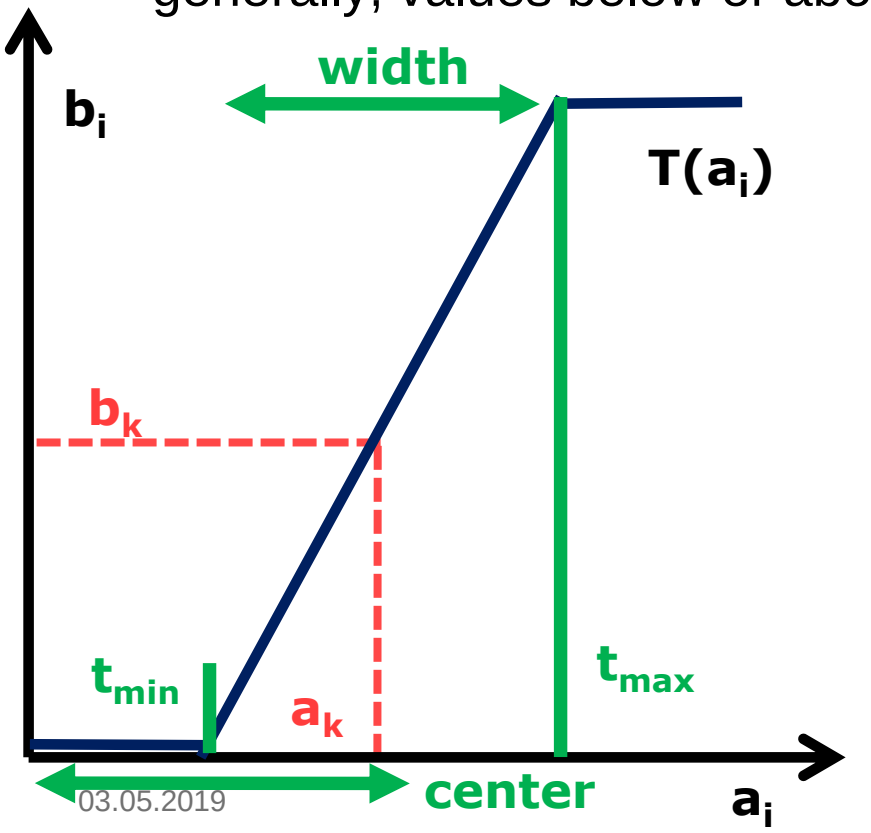
$T(a_i)$





Intensity Transformations cont'd

- **windowing**: increase contrast in relevant intensity range
  - all scalar values of input image ***A*** within interval threshold sub-range $[t_{min}; t_{max}]$ defined by *centre/width* are extended to full scalar range of the result image ***B*** utilizing linear interpolation.
  - generally, values below or above the interval threshold are assigned a_{min} and a_{max} respectively.



$$T(a_i) := \begin{cases} a_{\min} & a_i < t_{\min} \\ \dfrac{a_{\max} - a_{\min}}{t_{\max} - t_{\min}} \cdot (a_i - t_{\min}) & t_{\min} \le a_i \le t_{\max} \\ a_{\max} & a_i > t_{\max} \end{cases}$$

[0;255] full range | ~[140;210] | ~[57.5;80.5]

Intensity Transformations cont'd

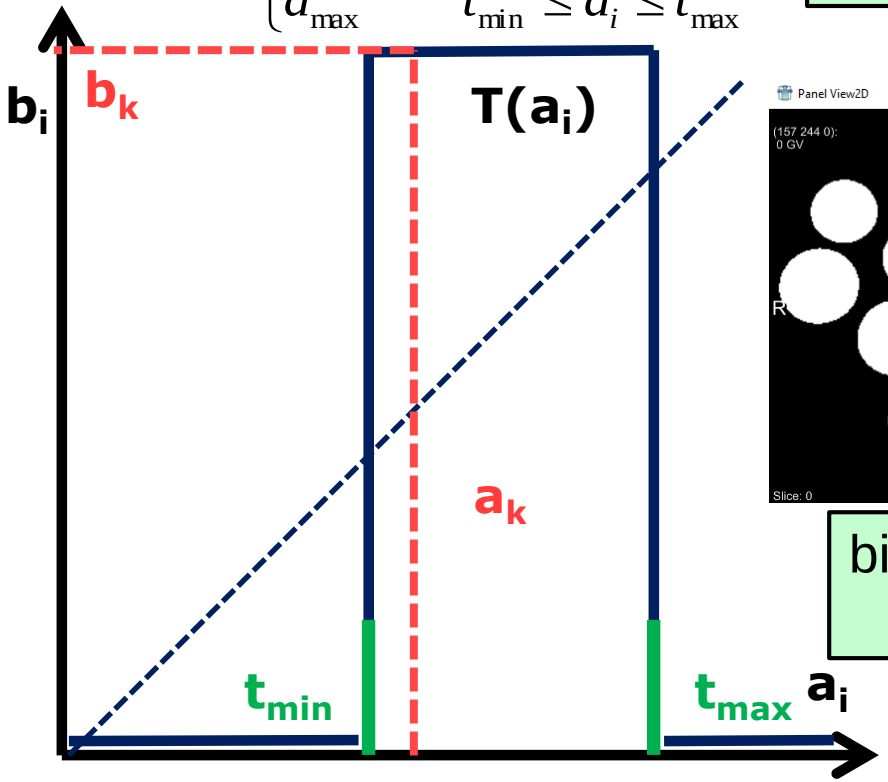
- **thresholding**: similar to windowing utilizing interval threshold sub-range $[t_{min}; t_{max}]$, with either conserved intensities within the window or binary result

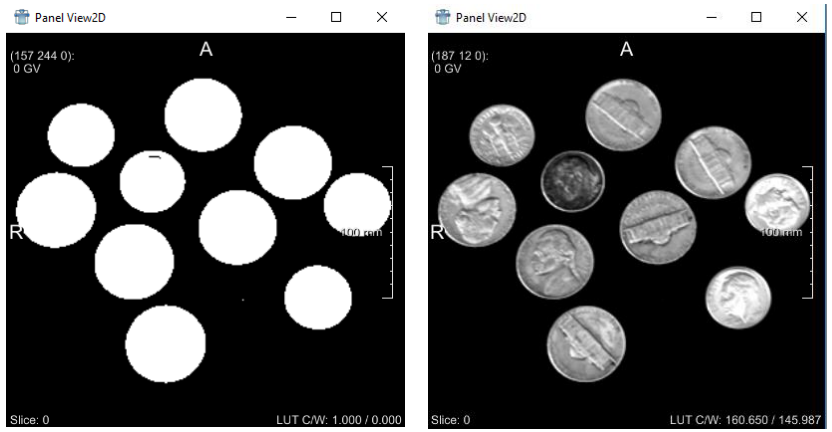
$$T(a_i) := \begin{cases} a_{\min} & a_i < t_{\min} \vee a_i > t_{\max} \\ a_{\max} & t_{\min} \le a_i \le t_{\max} \end{cases}$$

binary threshold

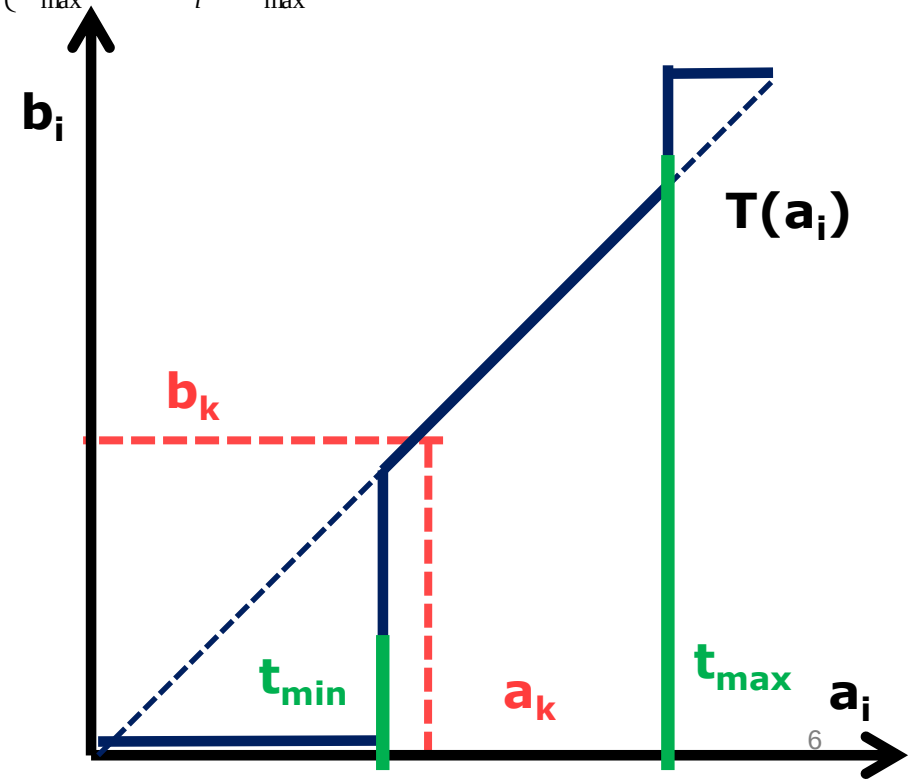
$$T(a_i) := \begin{cases} a_{\min} & a_i < t_{\min} \\ a_i & t_{\min} \le a_i \le t_{\max} \\ a_{\max} & a_i > t_{\max} \end{cases}$$

interval threshold



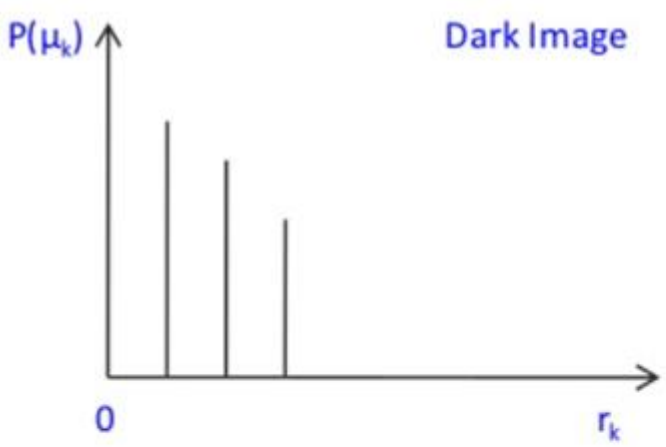


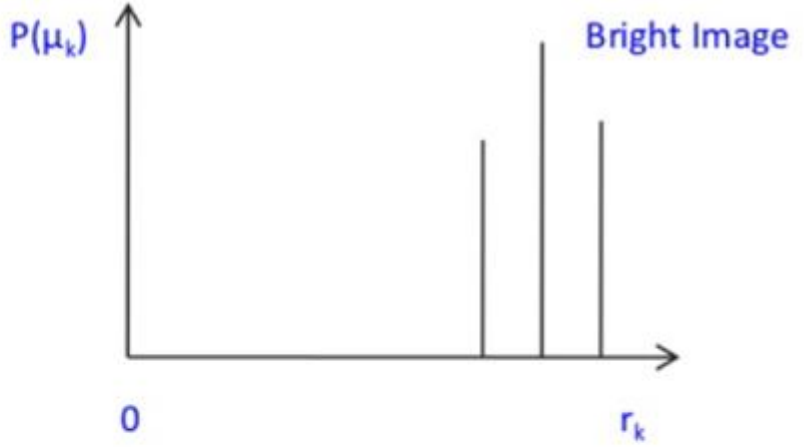
binary and interval threshold result in range [90;255]

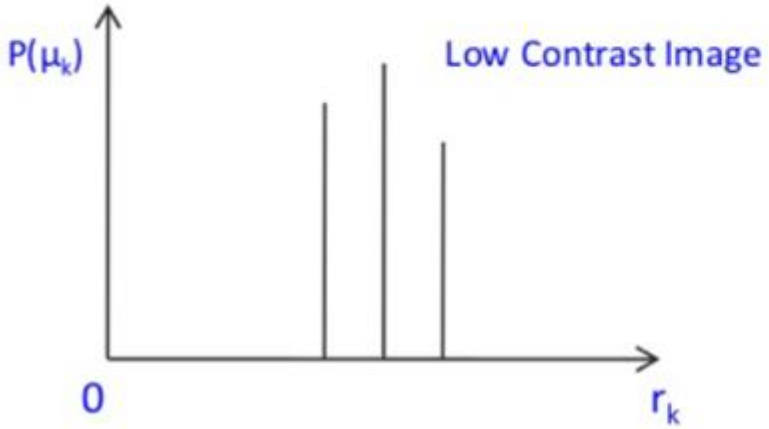


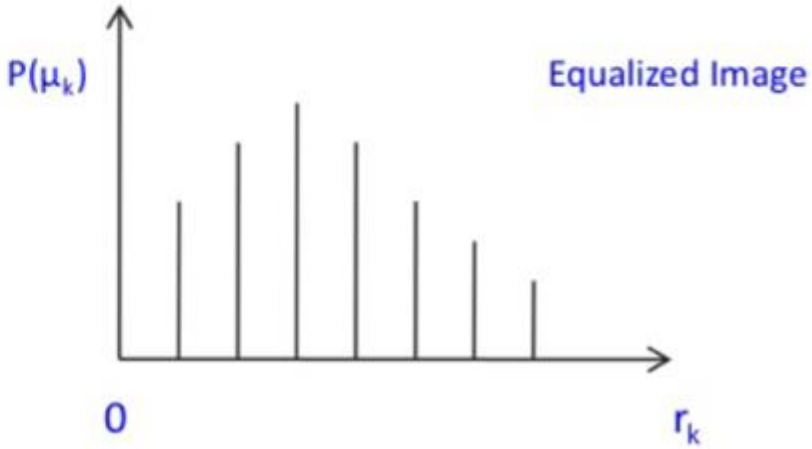
Intensity Transformations cont'd

- histogram as indicator for image contrast



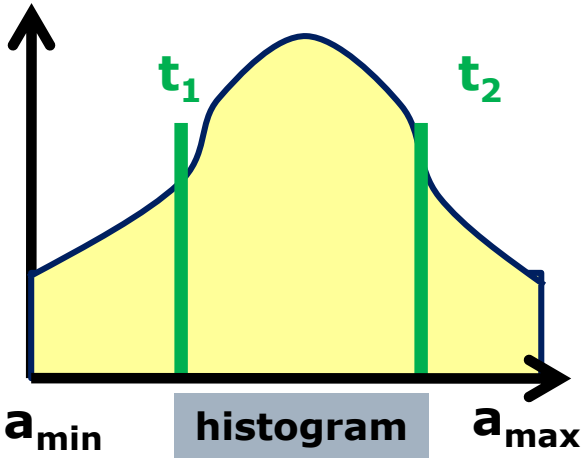


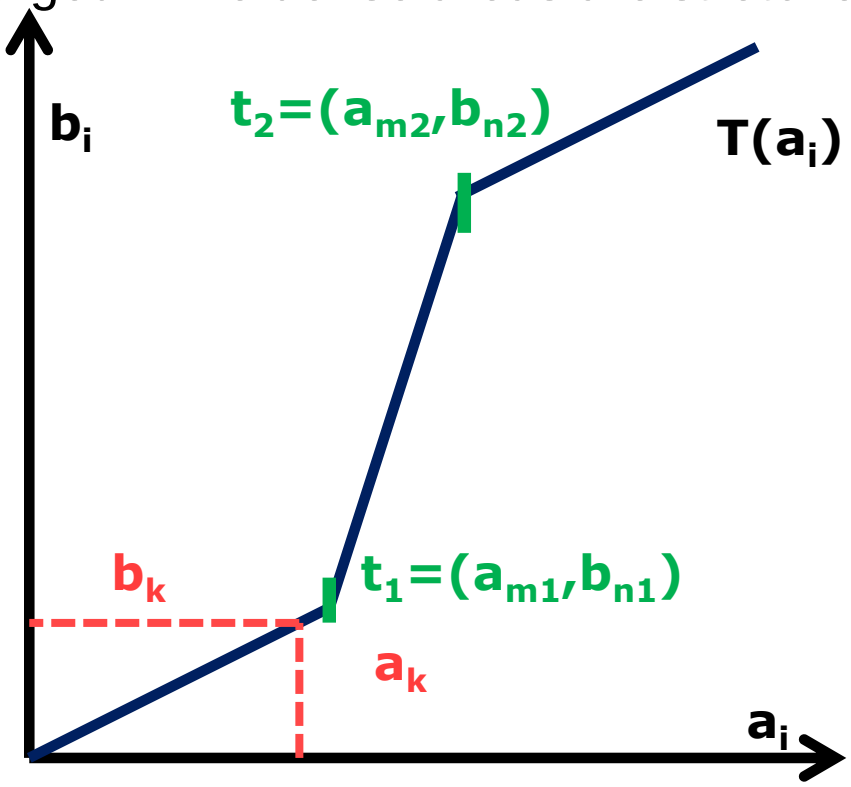




Intensity Transformations cont'd

- *selective contrast enhancement*, histogram equalization
  - image intensities are never equally balanced. Thus, in areas of sparse histogram values, the optical contrast (*granularity the human eye can discriminate*) is not utilized in an efficient way.
    - transfer function necessitates histogram besides scalar values
    - sparse areas are merged while dense areas are stretched





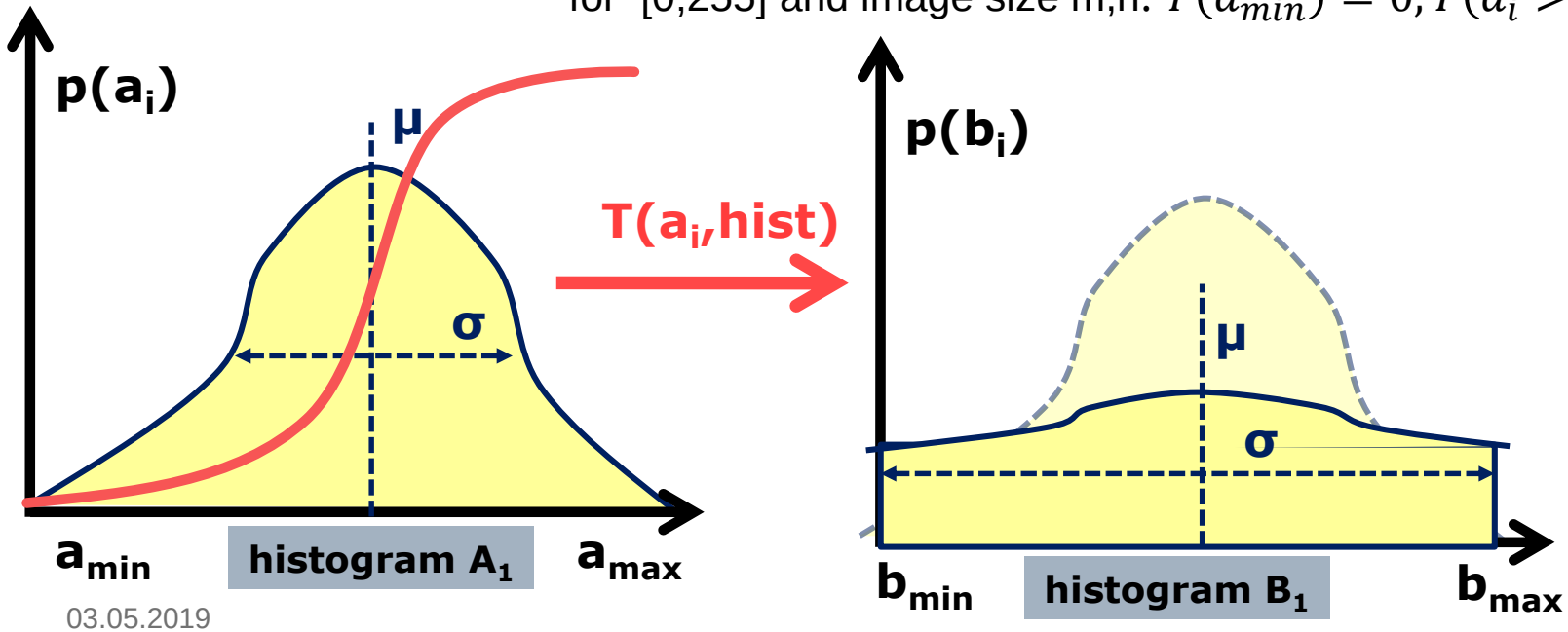
input image A

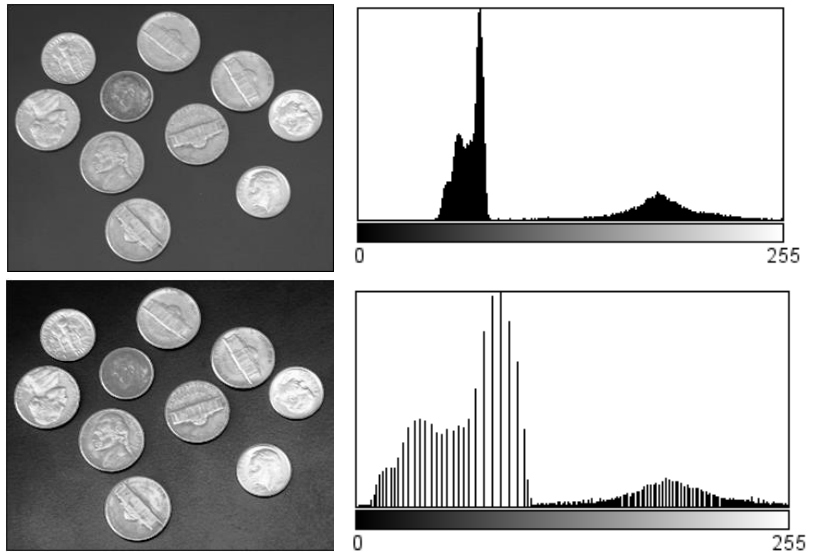
A' after contrast enhancement

Intensity Transformations cont'd

- histogram equalization strategy for automated contrast enhancement
  - non-linear intensity transform, enhancing the local contrast according to frequency of the particular intensities derived from histogram distribution
  - goal: uniform distribution of the intensities w.r.t. cumulated density function
    - if intermediate result is above the target, *bins* in B are left out (white holes in new histogram)
    - if intermediate result is below the target, several bins of A are merged for B.

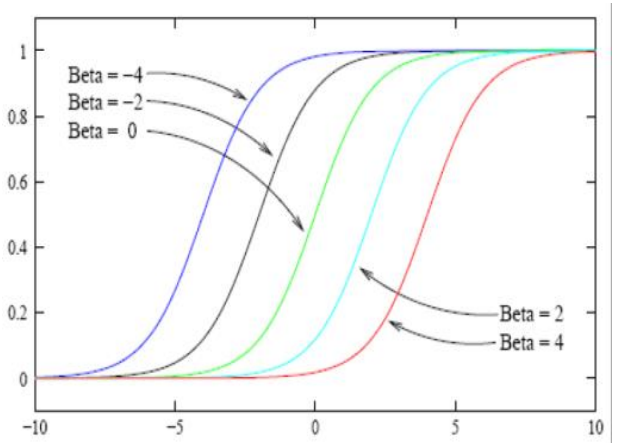
for [0;255] and image size m,n: $T(a_{min}) = 0, T(a_i > a_{min}) = round\left(\frac{\sum_{k=a_{min}+1}^{i} count(a_k)}{m \cdot n - count(a_{min})} \cdot 255\right)$



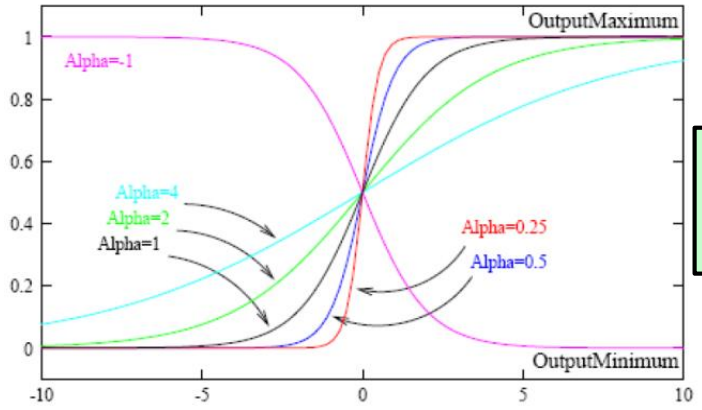


Intensity Transformations cont'd

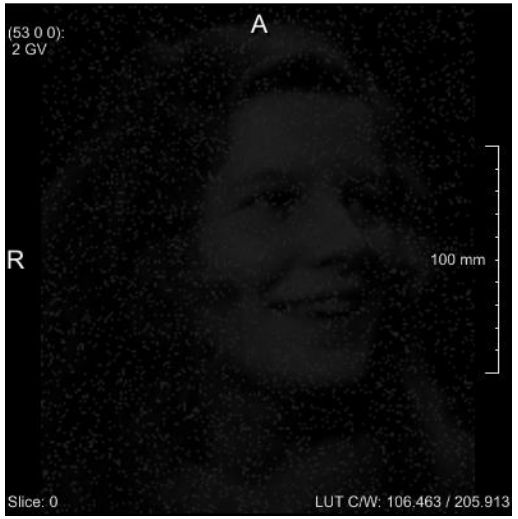
- sigmoid filter with (α, β) for generic contrast adaption/enhancement w.r.t mean intensity distribution (μ, σ).
  - α thereby steers the steepness of the transform, while β the position according to μ within range



effect of varying β for fixed α=1



effect of varying α for fixed β=1



orig image

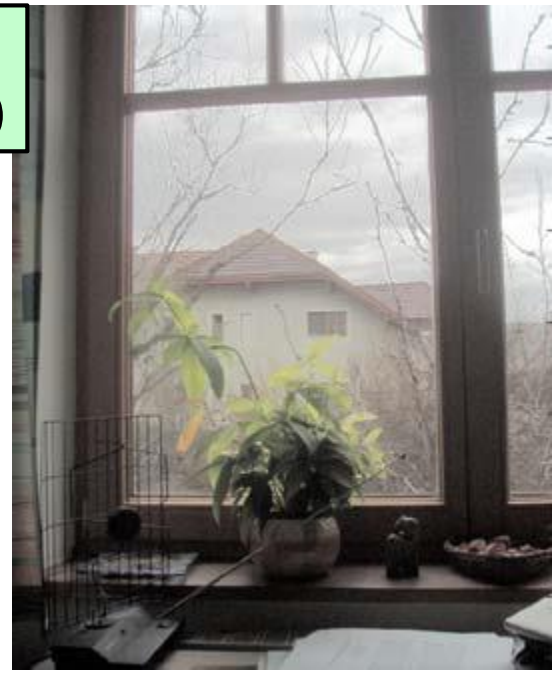
signoid with $\alpha = 6, \beta = 3$

Intensity Transformations on Colour Images

- applying intensity transformations for the colour channels in a separate way does not lead to sufficient results in case of RGB or CMY colour model
  - contrast in colour images is significantly depending on luminosity
  - thus, colour models like HSV or YUV should be utilized instead, thereby conserving the chromaticity while performing the intensity transformation

contrast enhancement applied for all RGB channels

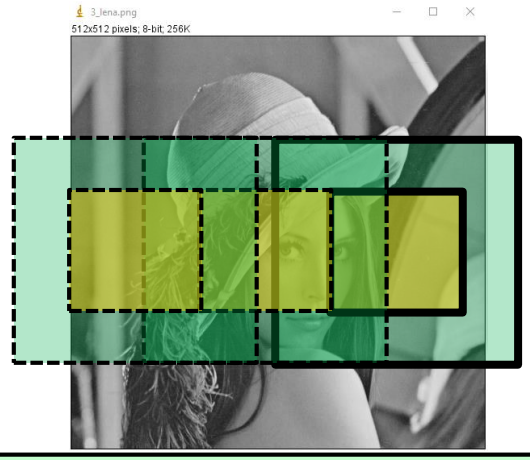
contrast enhancement applied for Y-channel only (YUC colour model)



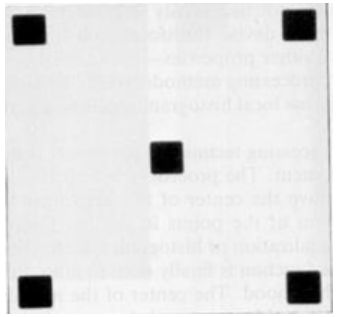
Adaptive Contrast Enhancement

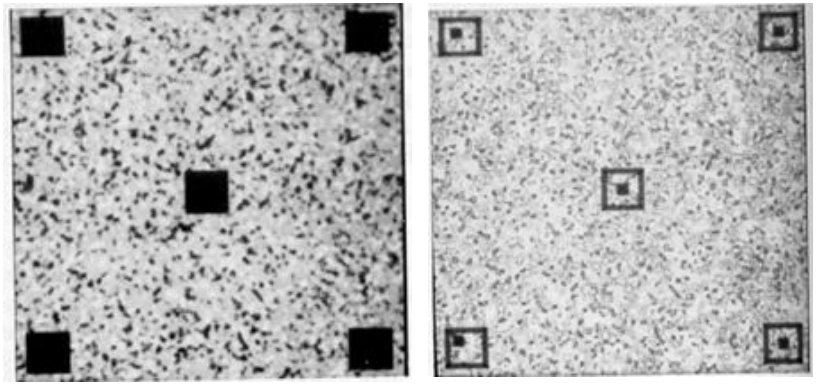
- in case of images with inhomogeneous illumination, globally applied scalar intensity transformations might lead to insufficient results.
  - thus, processing has to be performed for small sliding windows iterating over the entire image to locally perform the intensity transformation in the particular ROI (region of interest)
  - to prevent from artefacts and sharp breaking lines at the ROI borders, the sliding windows should be applied in an overlapping manner.

non-overlapping yellow ROIs



yellow ROI in target image B is calculated from larger surrounding ROI in A (green)





input A with global (*lower left*) and adaptive (*lower right*) contrast enhancement

Overview

- for two images I, M of matching size $width(I) == width(M) \wedge height(I) == height(M)$ any logical or arithmetic operation can be applied as binary operation in a pixel-wise manner.
  - logical operations (in case of binary images $[0; 1]$): **AND**, **OR**, **XOR**, **NOT**
  - arithmetic operations: **add**(+), **subtract**(-), **mult**(*), **div**(/)
  - operations combined to perform masking on the images
- further relevant:
  - unary operations like absolute value **abs** or difference **diff**
  - unary operations by using a constant instead of the second input image

Image Operations - Overview

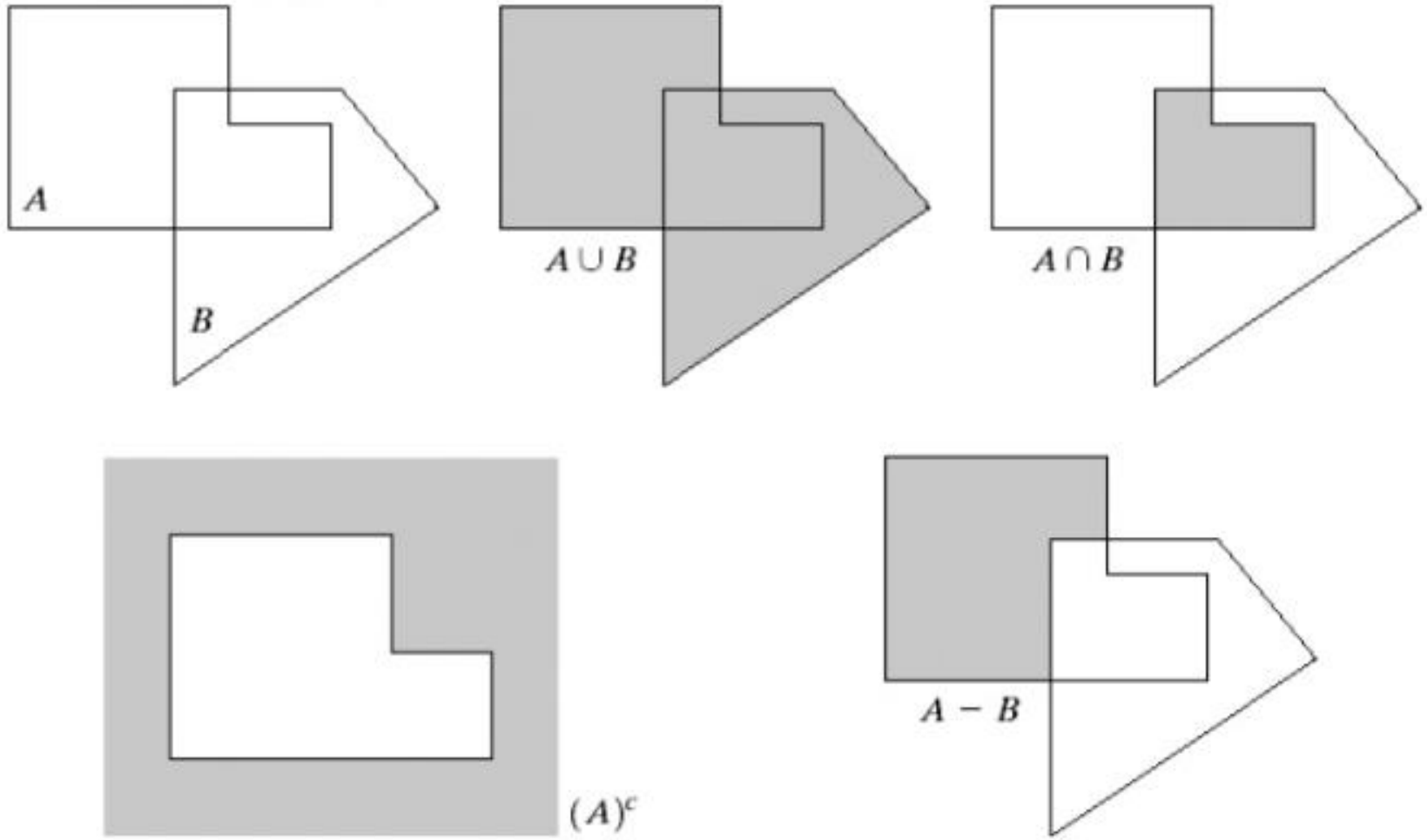
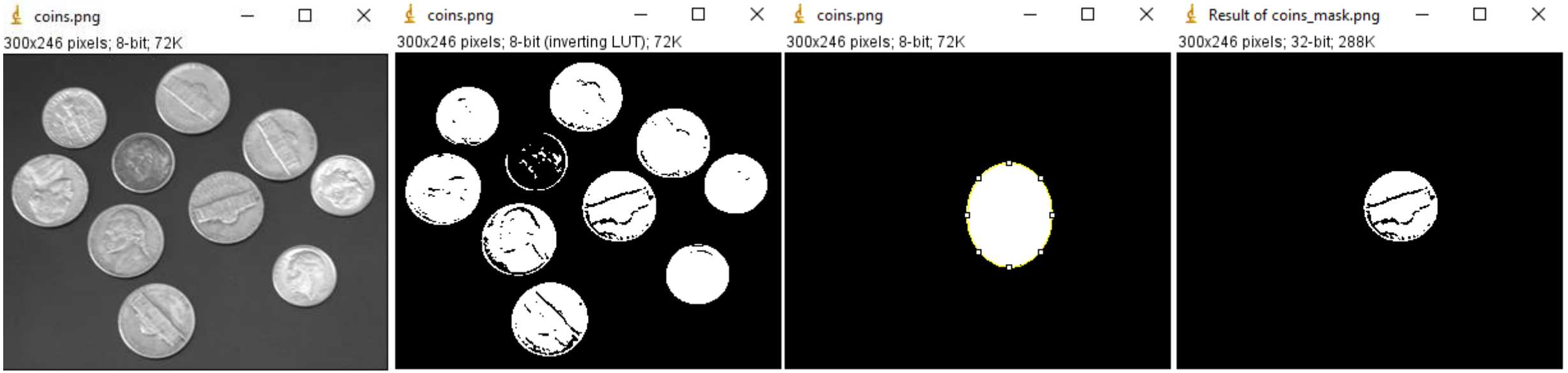


Image Operations - Examples

- **masking**: applying larger, e.g. manually set mask, on over-segmented image data with **AND** operation



original image

over-segmented image

manually drawn mask on target image

result of masking

Image Operations - Examples

- operation **DIFF** relevant for visually evaluating the filter effect

original image

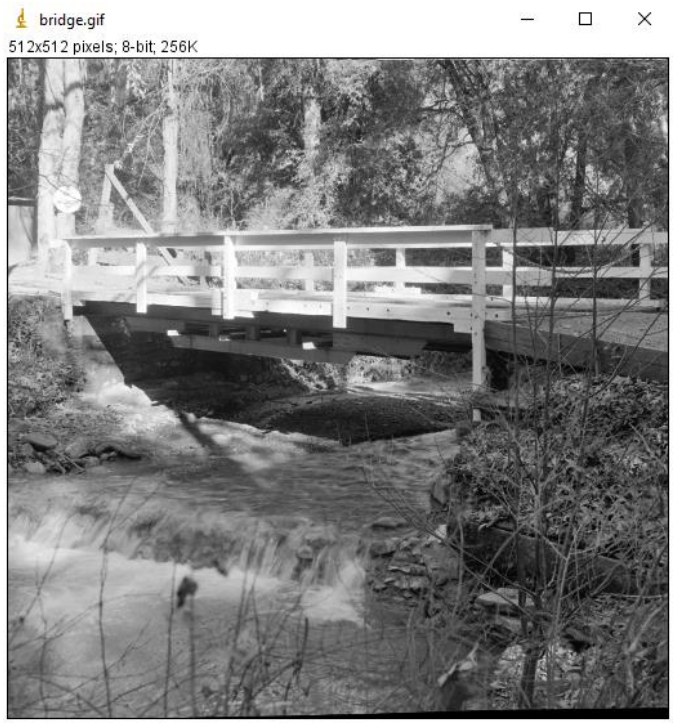
image after massive blur

difference image

Image Operations - Examples

- operation **ADD** and **DIV** to calculate a weighted average image, c.f. *morphing*

A, Original

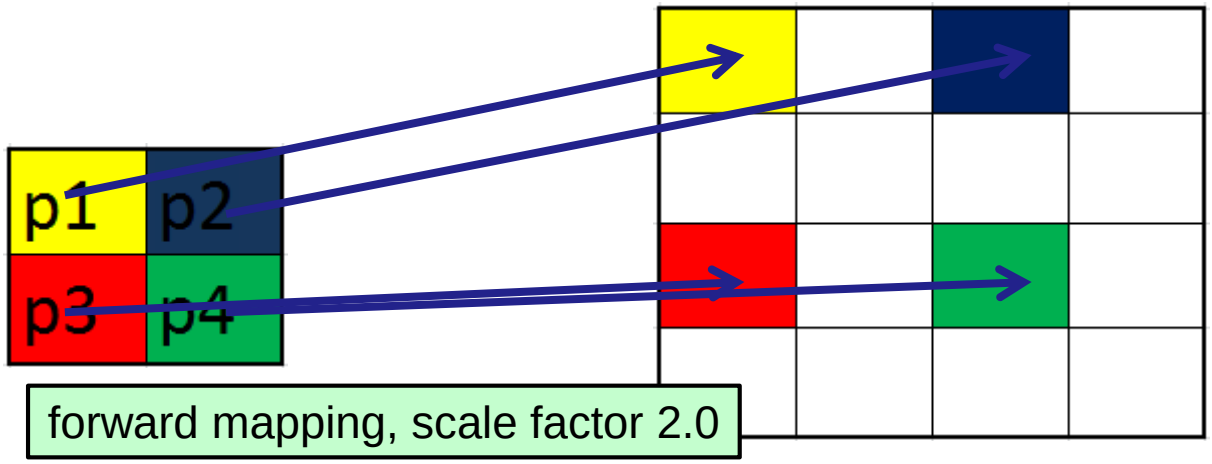


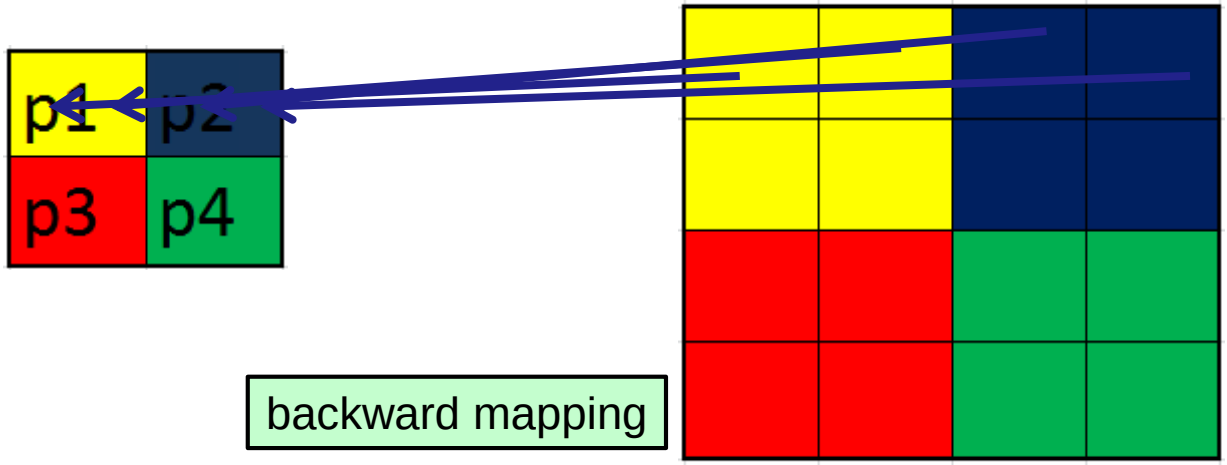
B, Original

(A+B)/2

Resampling

- if changing size of an image ***A***, the available information needs to be re-sampled at varied discrete image raster to fill up pixel values for result image ***B***.
  - thereby *backward-mapping* is favourable to guarantee
  - if scaling up by a factor of e.g. $s = 3.14$, sampling lacks exactly matching pixel positions to access the values in ***A***.
  - Consequently, interpolation is applied to calculate the intermediate values beyond discrete pixel raster





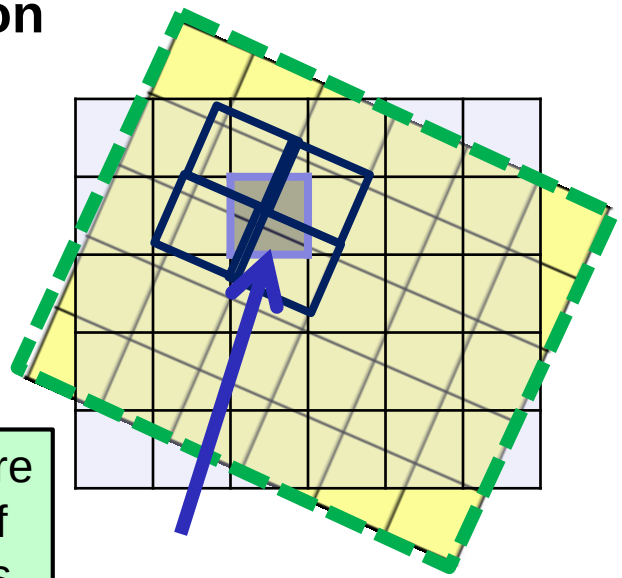
Resampling cont'd

- pixels can be interpreted as small squares assigned a constant scalar value that can be accessed via indices exactly in the mid.
- in the example below, image A can be evaluated at discrete positions $A(2,2) = 1$, $A(3,2) = 1$, $A(2,3) = 1$ and $A(3,3) = 2$.
  - but what is the value at intermediate position $A(2.5, 2.7)$?
  - intermediate positions need to be evaluated when resampling (scale) or transforming (translation, rotation) the image data ➔ **interpolation**

(0,0)

0	1	0	3	4	5
0	0	0	1	1	2
0	1	1	1	3	2
0	1	1	2	2	3
1	7	6	5	4	3

what is the correct value at position A(2.5, 2.7)?



neighbouring pixel values are evaluated for calculation of intermediate pixel positions

Interpolation

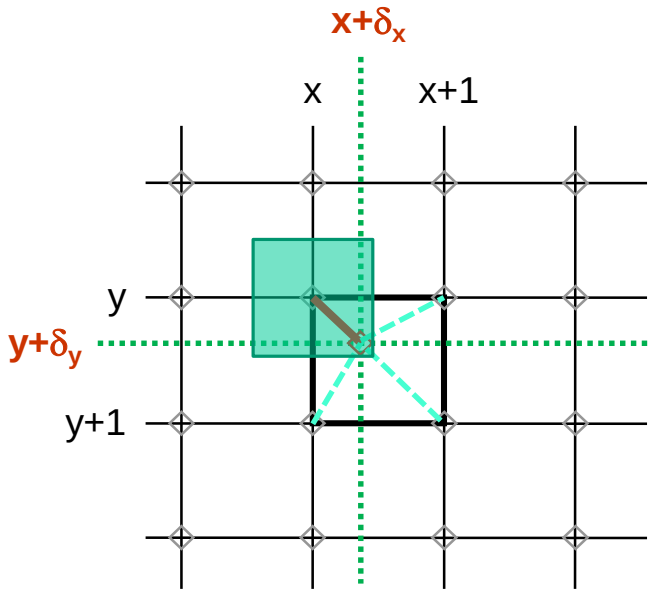
- in case of resampling the image, e.g. reducing size to 2/3, interpolation strategies are required to calculate the scalar values for the intermediate positions beyond the discrete image grid
- interpolation strategies:
  - *nearest neighbour interpolation*
  - *bilinear interpolation*
  - *cubic interpolation*

image A, 100%

image B, 66%; each 1.5th pixel

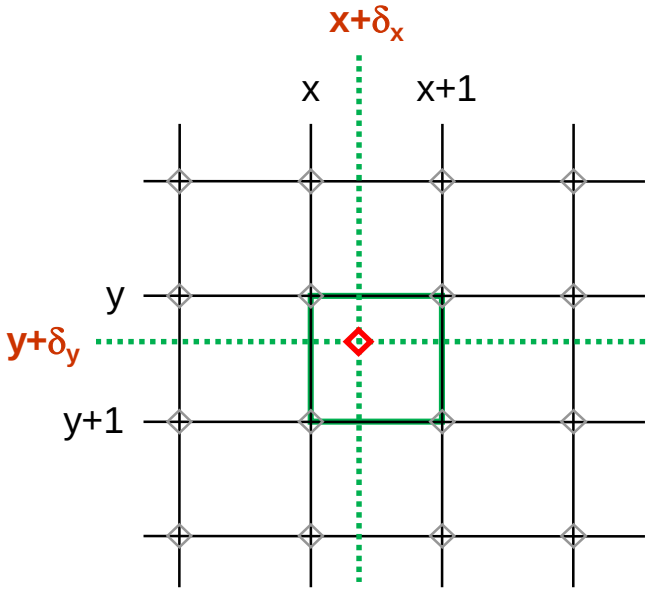
Interpolation

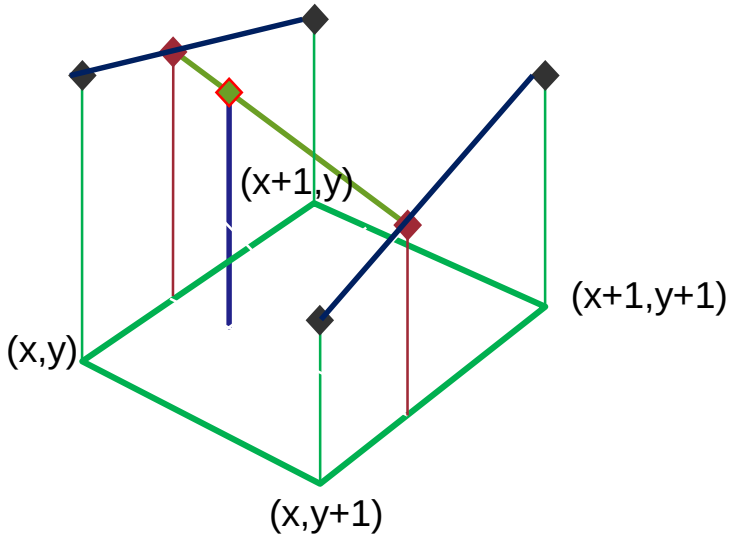
- Nearest Neighbour Interpolation
  - pixel in **A** closest to the target position contributes with its value for assignment in **B**
  - same as „rounding“ the coordinates in an arithmetic way
  - very high performance but low quality (blocks of identical scalar values, aliasing,...)



Interpolation

- Bilinear Interpolation
  - weighted average of the surrounding 4 pixel positions
  - the lower the linear distance to the target coordinate, the higher the influence on the resulting scalar value
  - 3 linear interpolations, 2 times in x direction and finally one time in y-direction

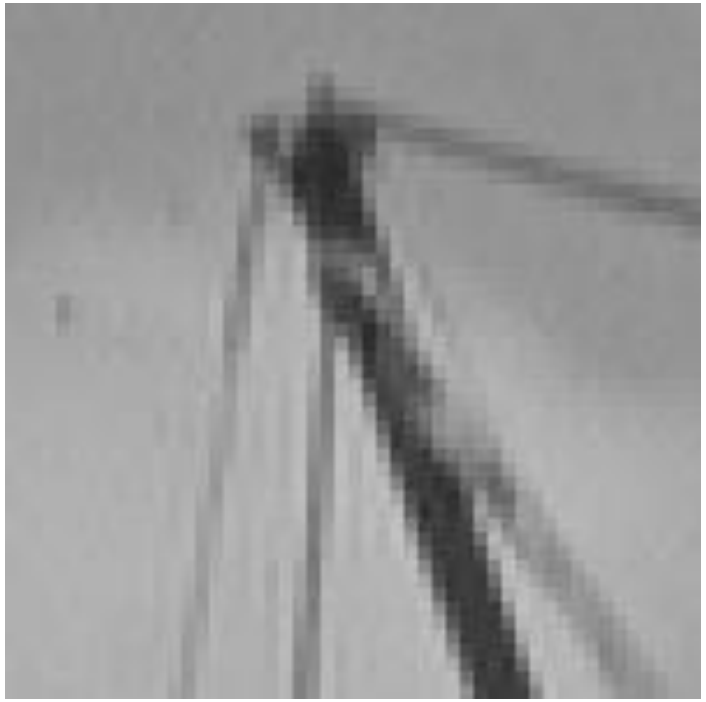




$$g(x+\delta_x, y+\delta_y) = (1-\delta_x)(1-\delta_y)g(x,y) + \delta_x(1-\delta_y)g(x+1,y)$$
$$+(1-\delta_x)\delta_y g(x,y+1) + \delta_x\delta_y g(x+1,y+1)$$

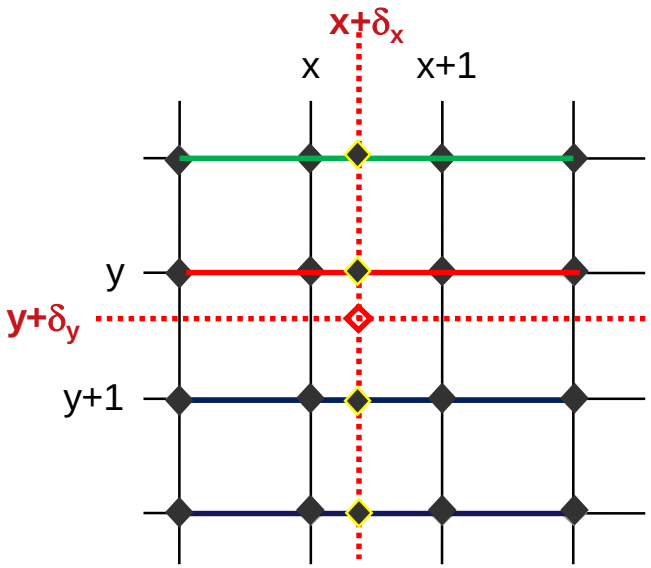
Interpolation

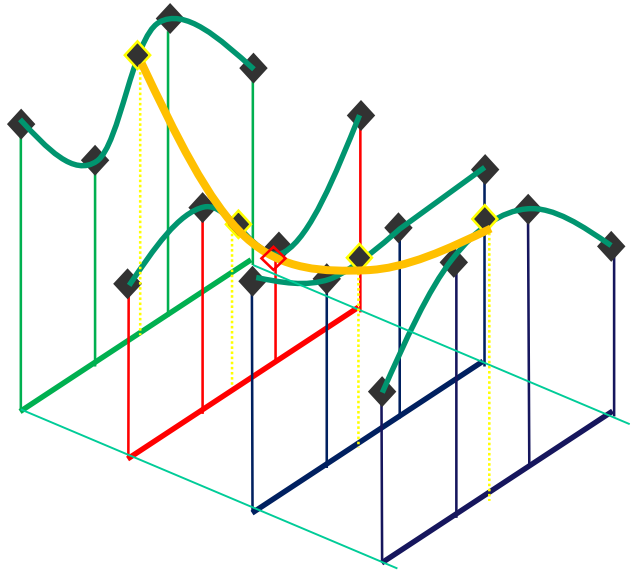
- comparing results of NN and Bilinear Interpolation, significant differences arise.
  - NN interpolation leads to sawtooth-shaped edges while Bilinear interpolation ensures smooth edges



Interpolation

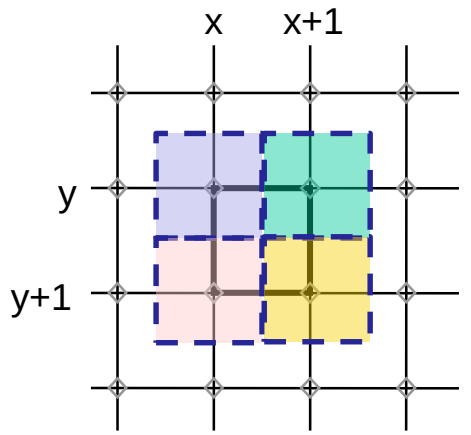
- cubic interpolation
  - Splines, i.e. multi-dimensional polylines are fit into scalar topography in local neighbourhood
  - thus, arbitrary intermediate pixel-coordinate positions can be evaluated
  - $y = a_1 + a_2 * x + a_3 * x^2 + \ldots + a_n * x^{n-1}$
  - e.g. bi-cubic incorporating 4x4 pixels





Coordinate Mapping

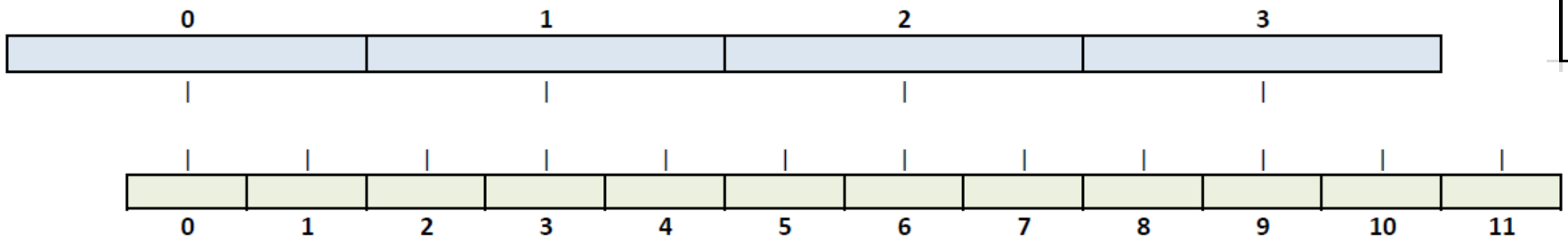
- Mapping of coordinates required in the course of resampling. Input image **A** with width w_a and result image **B** with altered width w_b.
- scale factor **s** as ratio of the image extents $s = w_b/w_a$
- all indices $[0..w_a-1]$ of image **A** need to be mapped to the indices $[0..w_b-1]$ of result image **B**
- reminder: a discrete voxel position, e.g. A(2,3), represents a rectangle of size (1,1) but the exact scalar value is only represented at the center. Positions besides the exact center necessitate interpolation strategies



Coordinate Mapping

- **a)** direct mapping calculated from scale factor **s**:
  - b = a * s
  - result will show an under- or over-represented at the right border
  - example: w_a=4, w_b=12, s=3.0

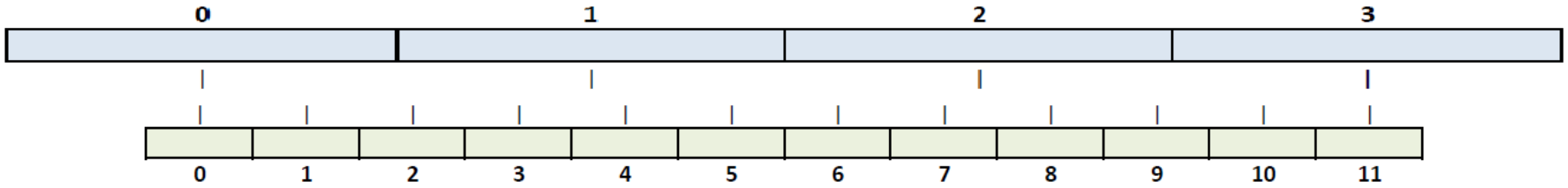
a	b
0	0
1	3
2	6
3	9



Coordinate Mapping

- **b)** a bit more balanced results will arise if the scale factor is adjusted with
  - s‘ = (w_b-1) / (w_a-1); b = a * s‘
  - example: w_a=4, w_b=12, s=3.0, s‘ = (12-1)/(4-1)=11/3=3.66
  - begin and end still under- or over-represented

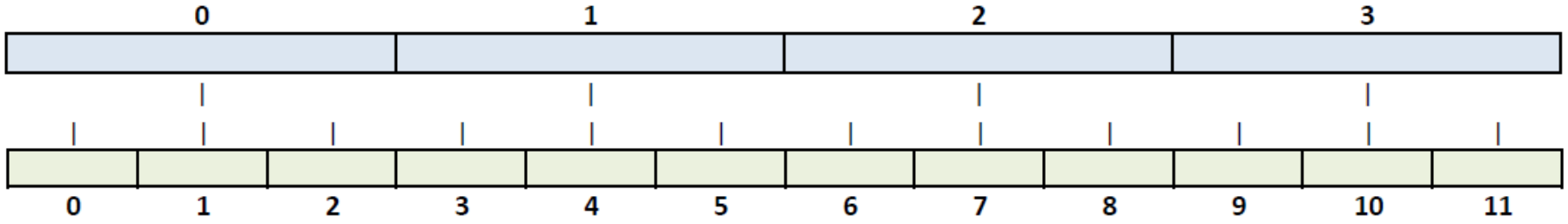
a	b
0	0
1	3,6666
2	7,3333
3	11



Coordinate Mapping

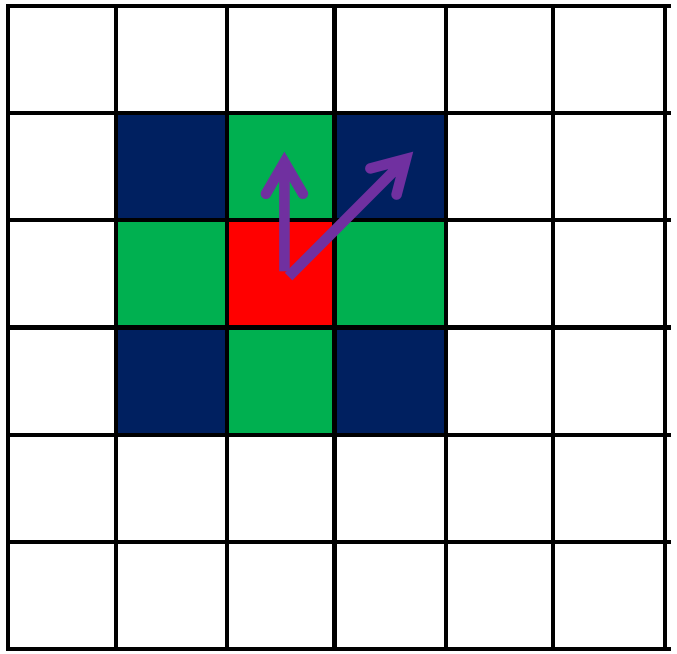
- **c)** perfect overlapping of the pixel positions
  - thereby the relative position r [0.0;1.0] within the range is calculated
  - furthermore, the pixel-width of 0.5 units is considered too
  - $r_a = 1/(2\mathbf{w_a}) + a/\mathbf{w_a}$; $b = (2r\mathbf{w_b} - 1) / 2$
  - BSP: $\mathbf{w_a}=4$, $\mathbf{w_b}=12$

a	r	b
0	0,125	1
1	0,375	4
2	0,625	7
3	0,875	10



Neighbourhood adjacency

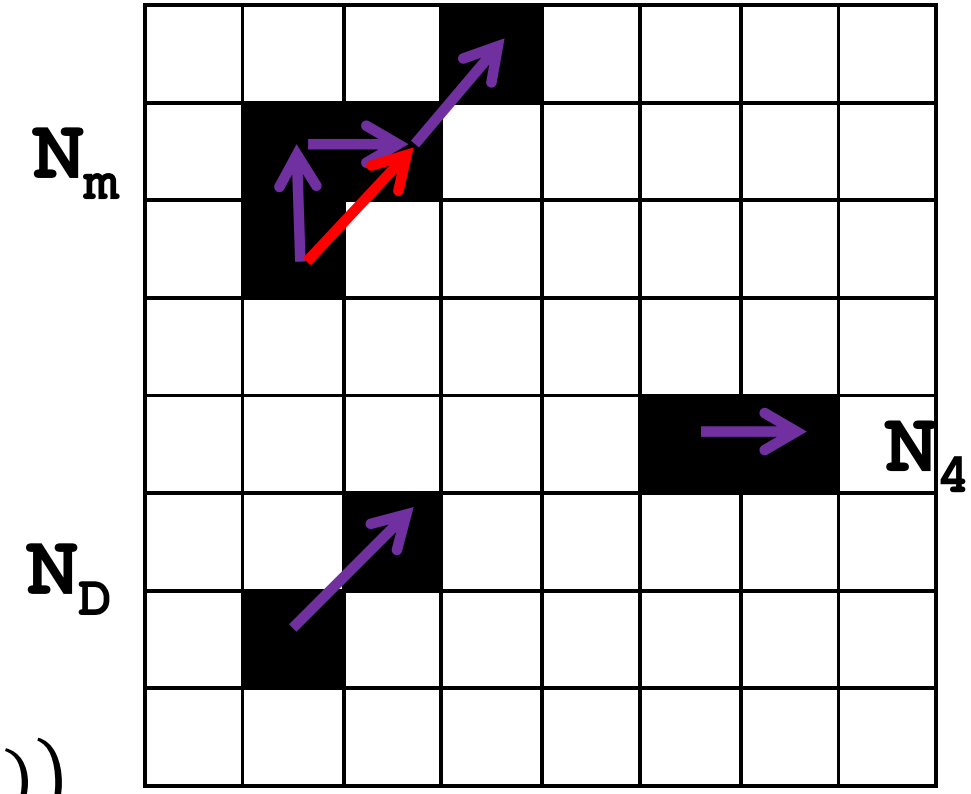
- a neighbourhood adjacency, i.e. connectivity and length of the path, can be defined for pixels of a discrete image matrix
- for pixel ***q*** (**red**) there are four direct neighbours in $\boldsymbol{N_4}$ (**green**), i.e. left, right, top, bottom
- the diagonal neighbours $\boldsymbol{N_D}$ (**blue**) show different values in both, *x* and *y* compared to the centre denoted as hot-spot.
- full direct neighbourhood for 2D pixels is denoted as $\boldsymbol{N_8}$ with $N_8 = N_4 \cup N_D$



Neighbourhood adjacency cont'd

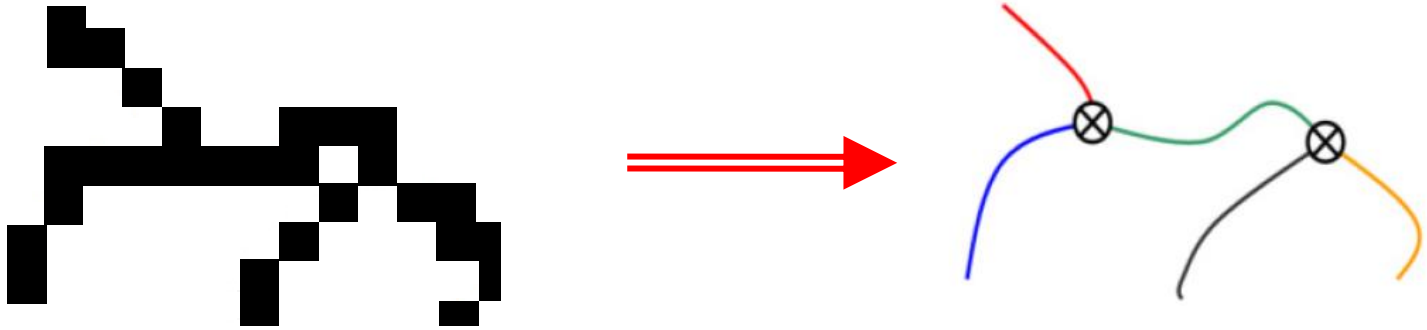
- for length and validity of the connection between two neighbouring pixels $p, q \in I, p \neq q$ one can define:
  - $N_4(p,q) := |p,q| = 1$
  - $N_D(p,q) := |p,q| = \sqrt{2}$ utilizing Euclidean distance
- and adjacencies are defined:
  - $N_8(p,q) := N_4(p,q) \mathrm{V} N_D(p,q)$
  - mixed adjacency N_m to guarantee unambiguous paths as direct connections are favoured over diagonal ones with

$$N_m(p,q) := N_4(p,q) \mathrm{V}\left(N_D(p,q) \wedge ! N_4(p,q)\right)$$

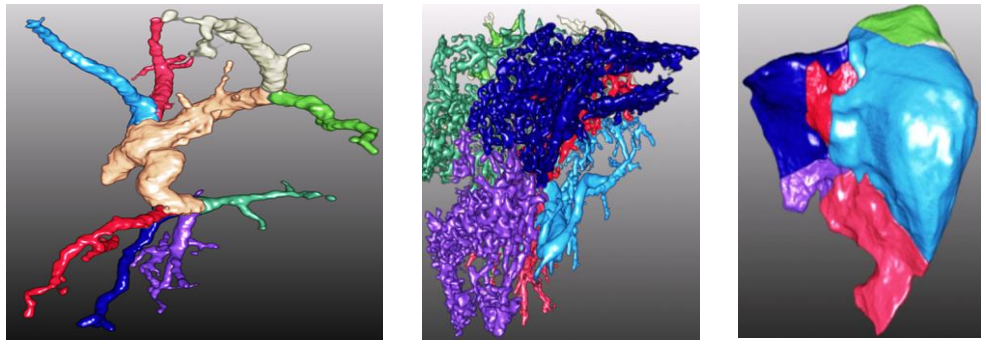


Neighbourhood adjacency cont'd

- mixed adjacency relevant for interpretation of graph structures (after segmentation and thinning denoted) with pixels in vectors (vectorization) interconnected by bifurcation points



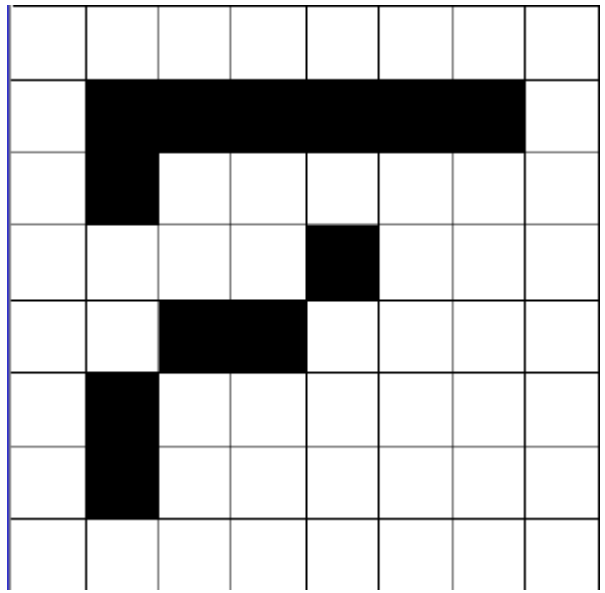
  - e.g. vessel hierarchy of liver in medicine



Neighbourhood adjacency cont'd

- definition of a *path*
  - a path between two pixels $p, q \in I, p \neq q$ exists for a specific neighbourhood adjacency, e.g. N_8, if the two pixels p, q are direct neighbours or if there exists a third pixel z, that defines a path to both, p and q (*triangle inequality*).
  - a path, i.e. connectivity, is given w.r.t. N_8 as

$$conn_{N_8}(p,q) := N_8(p,q) \vee \left(\exists z \in I, z \neq q, z \neq p : conn_{N_8}(p,z) \wedge conn_{N_8}(q,z)\right)$$



two valid pathes in N_8

Neighbourhood adjacency cont'd

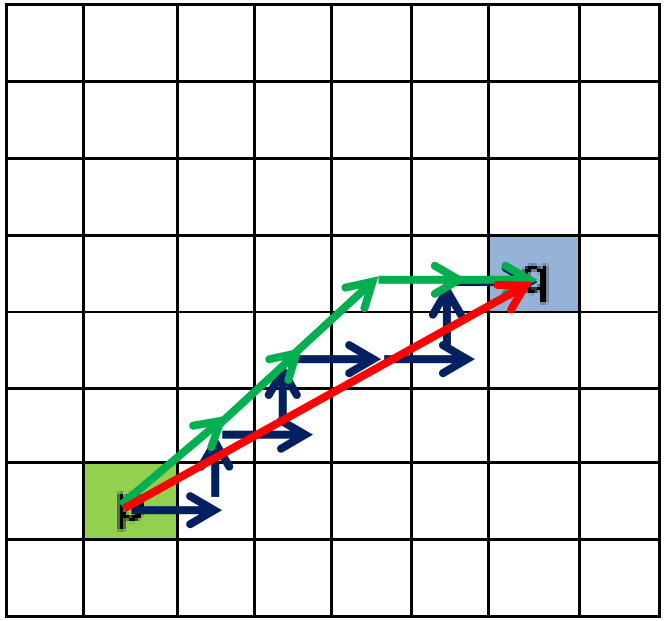
- distance for neighbourhood adjacencies is required to calculate the costs of an overall path between two pixels $p, q \in I, p \neq q$.
- several different *path metrics* exist (*Euclidean, Chess/Checker, Manhattan,...*) as elucidated in the following sections. Nevertheless, the following constraints must be fulfilled for all of the metrics:
  - exclusively a positive path length is permitted with $dist(p, q) \geq 0$
  - the triangle inequality must hold if considering the shortest paths in a graph only, thus $dist(p, q) \leq dist(p, z) + dist(q, z)$
  - calculation of a path length needs to be commutative, thus the order of sub-path aggregation needs to be invariant w.r.t. the overall result: $dist(p, q) == dist(q, p)$

Neighbourhood adjacency cont'd

- different distance metrics required for different requirements (*performance, accuracy,...*). While the direct connection distance in N_4 for two adjacent pixels p, q is always valued with 1, the diagonal can be valued utilizing different strategies:
  - Euclidean Distance: $dist(p,q) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}$
  - Manhattan Distance: $dist(p,q) = |x_p - x_q| + |y_p - y_q|$
  - Chess (Checker) Board Distance: $dist(p,q) = max(|x_p - x_q|, |y_p - y_q|)$
- the overall path utilizing e.g. N_8 adjacency is aggregated from the direct neighbour distances (c.f. mixed adjacency):
  - $D_{path}(\vec{P}) = \sum_{i=1}^{|\vec{P}|-1} dist(p_i, p_{i+1})$ with $\vec{P} = (p_1, p_2, \dots, p_n)$

Neighbourhood adjacency cont'd

- depending on the application domain, the practical distance metric needs to be chosen
  - e.g. for calculation of a distance map, Euclidean leads to the most accurate results
  - if processing a graph-grid, diagonal step might be of same costs as step in N_4, thus necessitating use of chess board distance
  - in the example to the right, the chosen distance metrics lead to different results:
    - Euclidean: **dist(p,q)** $= \sqrt{(1-6)^2+(6-3)^2}$ $= \sqrt{34} \simeq$ **5.831**
    - Manhattan: **dist(p,q)** $=$ **8**
    - Chess board: **dist(p,q)** $=$ **5**



Distance Map

- based on a chosen adjacency, a distance map can be calculated as image
- calculation of the distance map as iterative optimization problem
  - initially, all border elements are assigned a distance value of **0**, while all other pixels are assigned an undefined distance of ∞.
  - put all foreground elements onto a processing stack
    - take first element from stack: pixels might be updated, i.e. assigning a lower value, by checking the distances currently assigned to the direct neighbours.
    - weight of pixel *p* is updated, if $dist(q) + dist(p,q) \leq dist(p)$ with *p,q* in N_8.
    - any updated neighbour is put on the stack too, if not already on the stack
    - distance map calculation finishes if stack is empty
  - example: dist(p)=10, dist(q)= ∞, dist(p,q)=1 then after update dist(q)=11

∞	∞	∞	∞	∞	∞
0	**0**	**0**	**0**	**0**	∞
∞	∞	**0**	∞	∞	∞
0	**0**	**0**	∞	∞	∞
∞	∞	∞	∞	∞	∞

1	**1**	**1**	**1**	**1**	**2**
0	**0**	**0**	**0**	**0**	**1**
1	**1**	**0**	**1**	**1**	**2**
0	**0**	**0**	**1**	**2**	**3**
1	**1**	**1**	**2**	**3**	**4**

init and final distance map

Bibliography

- L. G. Roberts, “Machine Perception Of Three-Dimensional Solids,” Lincoln Laboratory, TR 315, MIT, Cambridge, 1963.
- A. Appel, “The notion of quantitative invisibility and the machine rendering of solids,” in *Proceedings of the 1967 22nd national conference on* -, 1967.
- J. Warnock, “A Hidden-Surface Algorithm for Computer Generated Halftone Pictures,” Computer Graphics Department, University of Utah, Salt Lake City, Technical Report TR 4-15, NTIS AD-753 671, 1969.
- G. S. Watkins, “A real time visible surface algorithm,” The University of Utah, 1970.
- E. E. Sutherland, R. F. Sproull, and R. A. Schumacker, “A Characterization of Ten Hidden-Surface Algorithms,” *ACM Computing Surveys*, vol. 6, no. 1, pp. 1–55, Jan. 1974.
- H. Gouraud, “Continuous Shading of Curved Surfaces,” *IEEE Transactions on Computers*, vol. C-20, no. 6, pp. 623–629, Jun. 1971.

UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

Image Processing

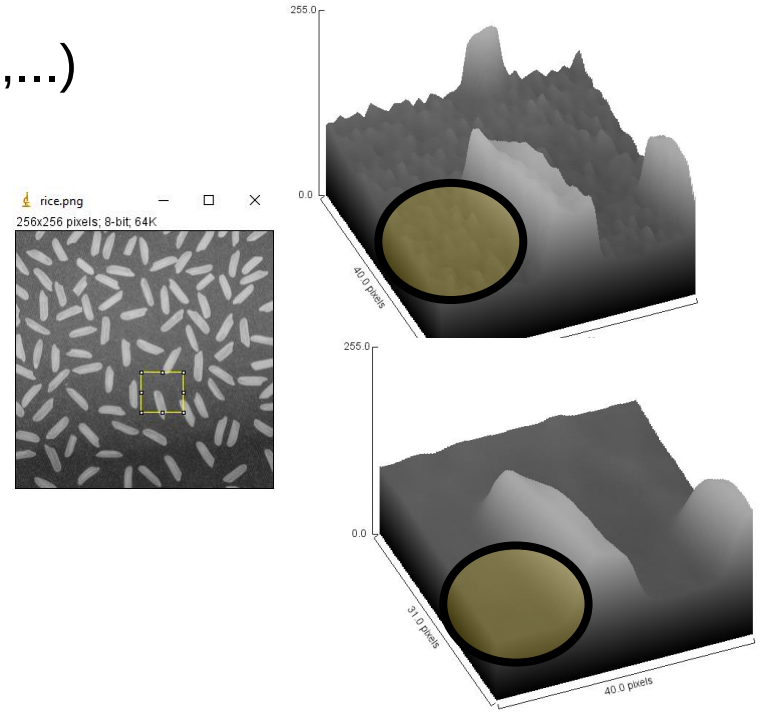
Image Filtering

DI(FH) Dr. Gerald Adam Zwettler, MSc, gerald.zwettler@fh-hagenberg.at
Lector of Computer Graphics, Image Processing and Computer Vision

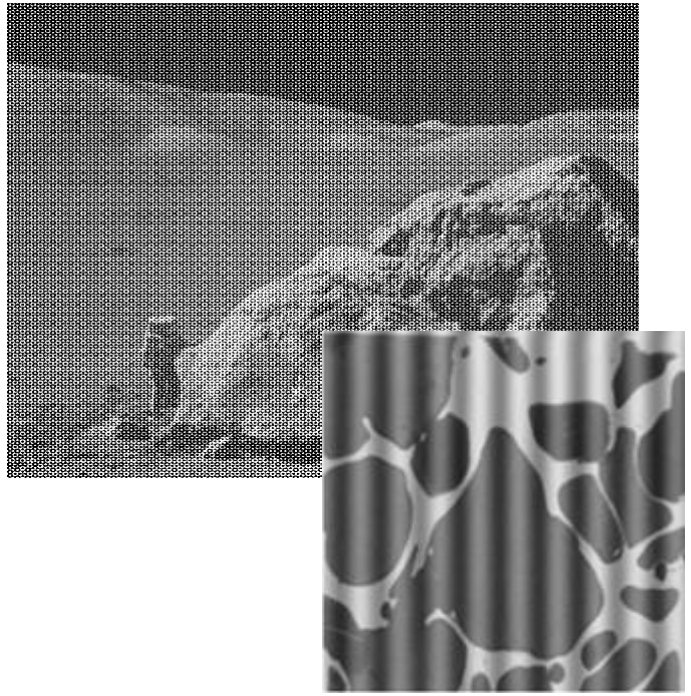
Image Filtering

- goals:
  - improve SNR (signal-to-noise-ratio) by applying smoothing
  - remove periodic noise
  - detect structures (lines, edges,...)

S&P (salt and pepper noise)



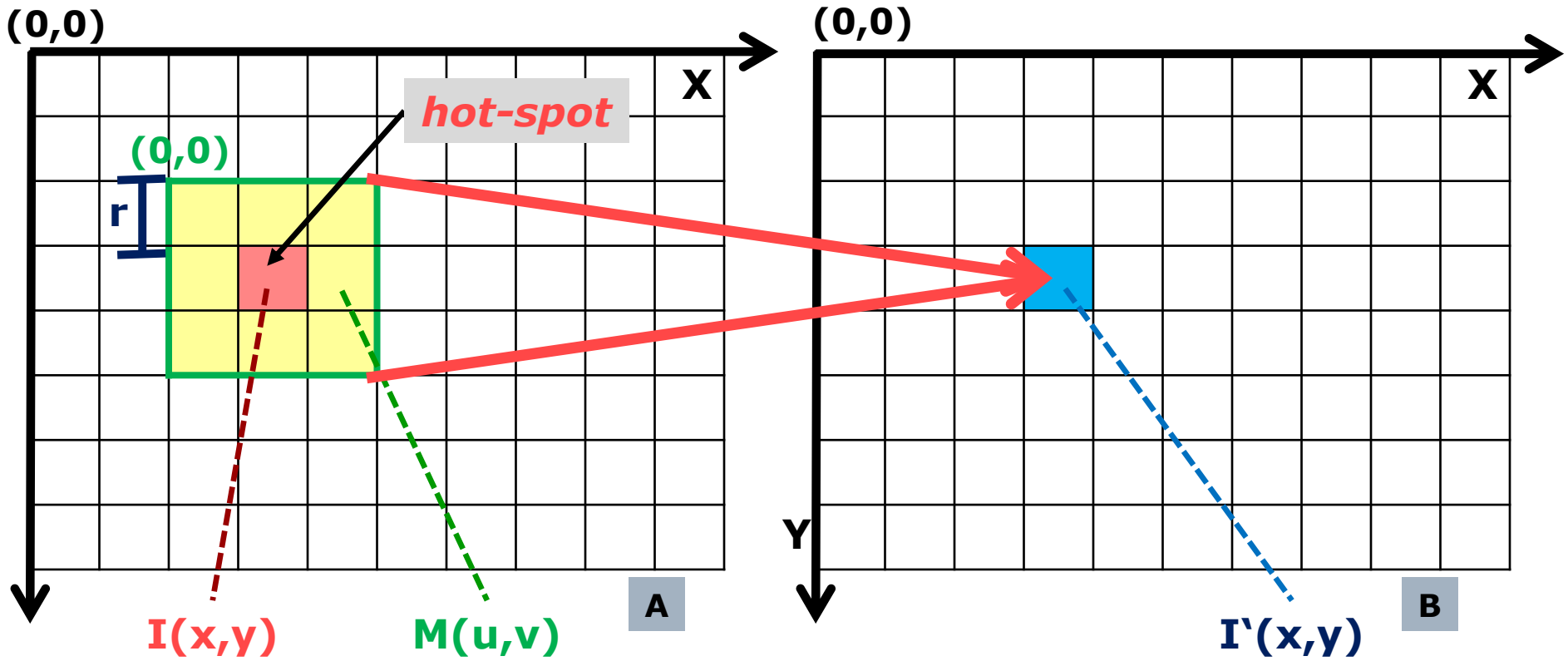
background inhomogeneities
(Guass-shaped noise distribution)



periodic noise
(frequency domain)

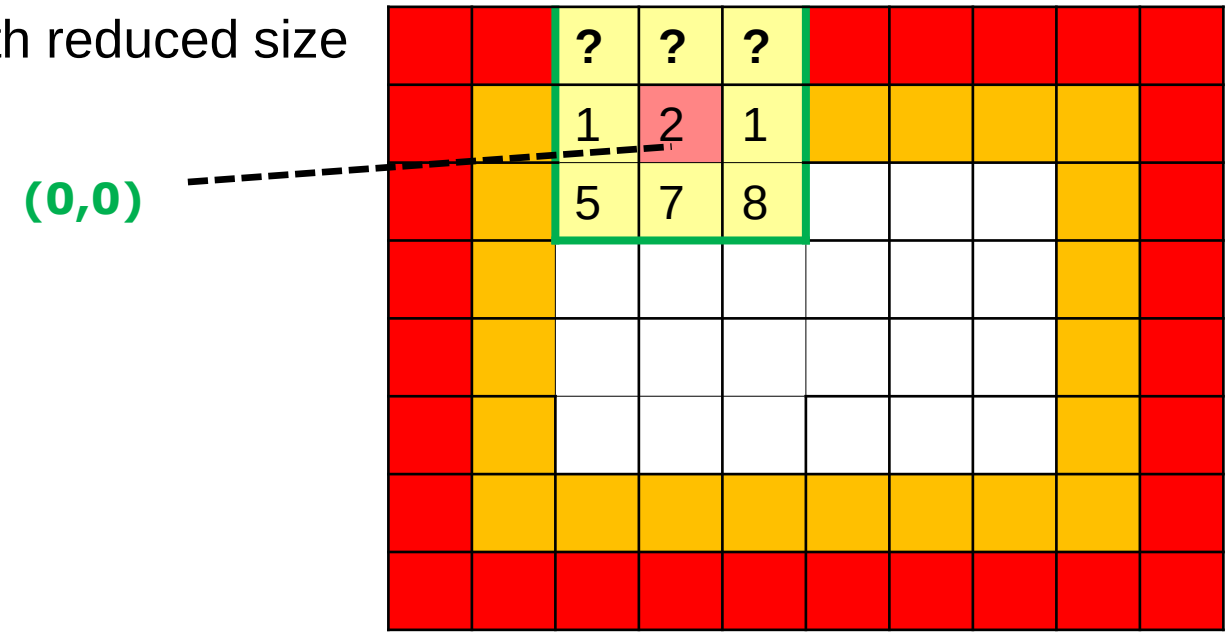
Filtering in Spatial Domain (*Ortsraum*)

- in spatial domain, convolution of input image ***I*** is applied utilizing a small mask ***M***
- the algorithm is always the same, only the mask design is relevant for the effect
  - a new pixel value $I'(x,y)$ in result image ***B*** is calculated in local neighbourhood of $I(x,y)$ in ***A*** by local multiplication with ***M*** within radius ***r*** and pixel-wise summing up the results



Filtering in Spatial Domain cont'd

- the convolution process $I \circ M$ is defined as $$I'(x,y) := \sum_{k=x-r}^{x+r} \sum_{l=y-r}^{y+r} I(k,l) \cdot M(k-(x-r), l-(y-r))$$
- in case of border areas, not all pixels are available for local multiplication in the course of convolution. Thus, three key strategies are available:
  - padding, i.e. enlarging input image A by an outer row of neutral scalar values w.r.t radius r.
  - checking the indices and leaving out the multiplication in case of accessing a value outside the defined pixel matrix I. In that case, normalization w.r.t. applied kernel sum might be required.
  - B with reduced size



in case of M showing r=1, the orange outer pixels of image A cannot be processed, as the red positions for multiplication are not given

Filtering in Spatial Domain cont'd

UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

- two categories of filter masks in spatial domain, namely:
  - **low-pass filters**
    - smoothing effect on input image.
    - High frequencies (fine details, edges) are damped, while the area information (low frequencies) is conserved
    - all mask elements in M sum up to **1**
  - **high-pass filters**
    - dampening of area information while conserving high frequencies
    - detectors for lines, points, gradients,...
    - all mask elements in M sum up to **0**.

Relevant Filter Kernels

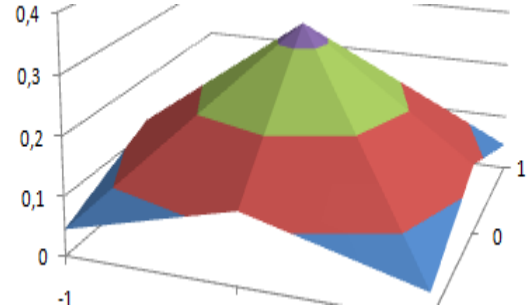
- ***low pass*** filtering, all exemplarily at radius r=1
  - mean convolution filter with equal weight for all kernel elements ➔ maximized smoothing effect

$$K_{mean} = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

  - gauss convolution filter with weights scaled according to distance from hot-spot. Regularly calculated from sigma ***σ*** to radius ***r*** ratio according to 2D gauss formula:

$$K_{gauss} = \begin{bmatrix} 1/18 & 2/18 & 1/18 \\ 2/18 & 6/18 & 2/18 \\ 1/18 & 2/18 & 1/18 \end{bmatrix}$$

$$K_{gauss}(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

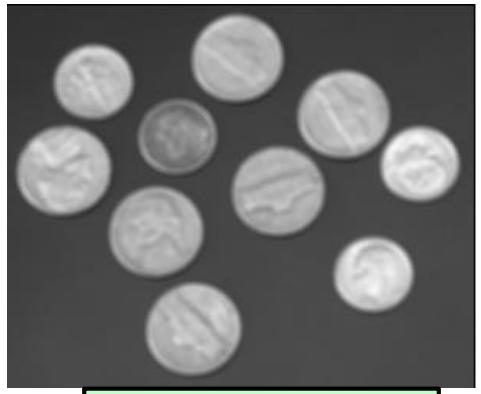


discrete approximation

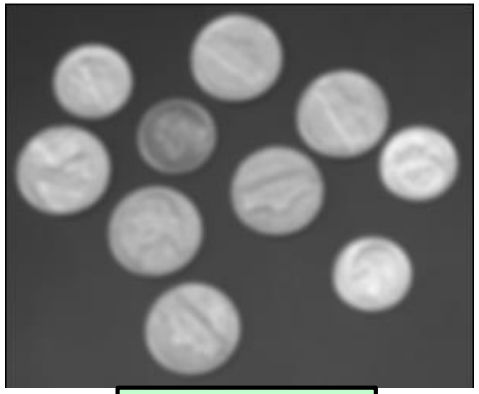
Relevant Filter Kernels cont'd

- **mean** kernel with significantly higher smoothing effect compared to **Gauss** kernel

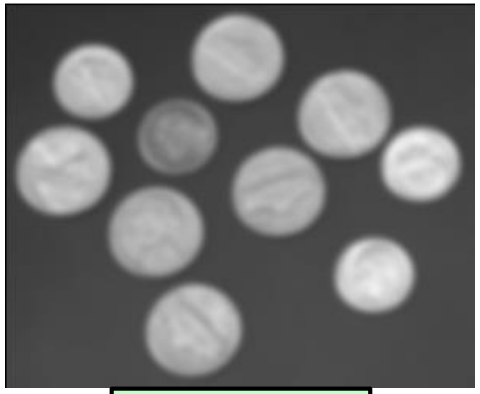
orig image



1x ***mean***, r=2

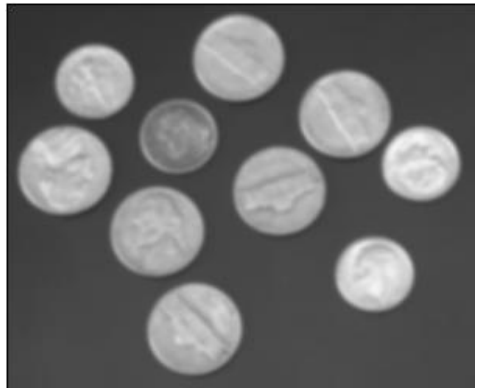


2x m, r=2

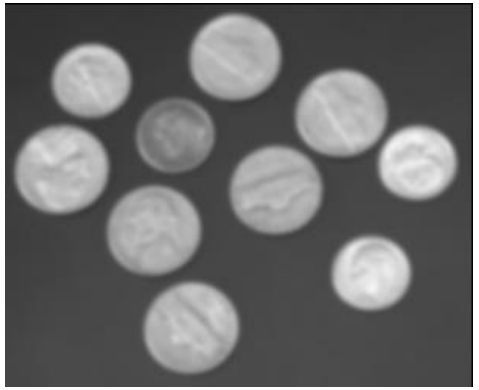


1x m, r=2

1x ***gauss***, r=2,σ=1



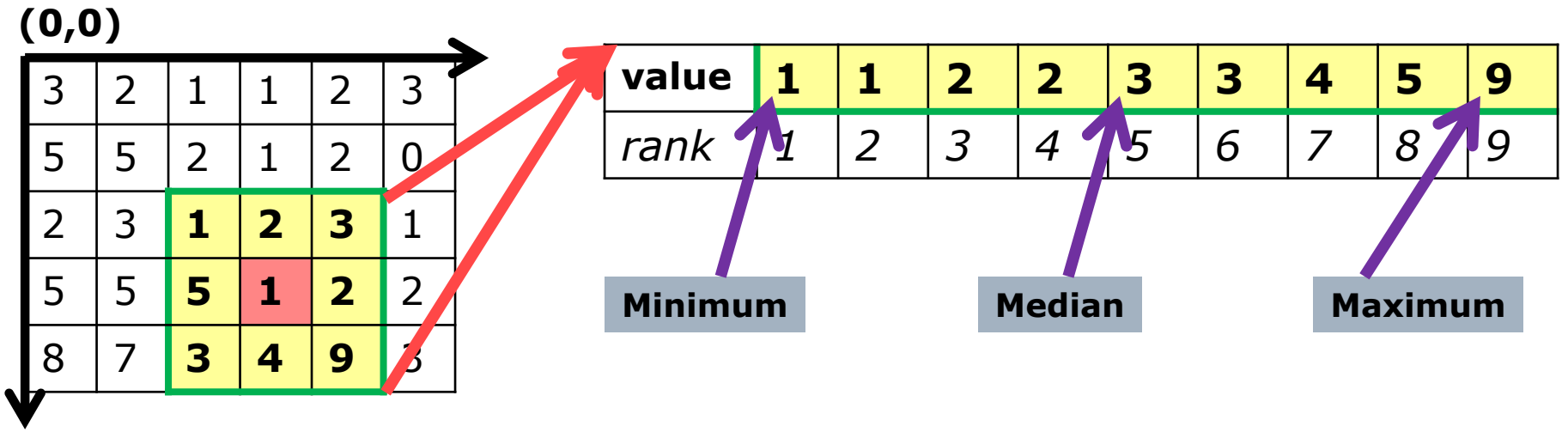
1x g, r=2,σ=1



1x g, r=2,σ=1

Relevant Filter Kernels cont'd

- **rank filters**: instead of utilizing filter mask ***M*** with specific coefficients, all scalar values of input image a at current mask position get sorted.
  - **Median filter**: assigning the midst value for result image B. In case of even element count, averaging of the two mid values is feasible. Perfectly applicable in case of salt and pepper noise
  - while Median filter shows a smoothing effect, **Minimum/Maximum** filters rather showing a detector effect



Relevant Filter Kernels cont'd

- rank filters cont'd:
  - perfect for removing S&P noise
  - flat areas and intensity increase/decrease is conserved, while outliers are eliminated and local minima/maxima flattened

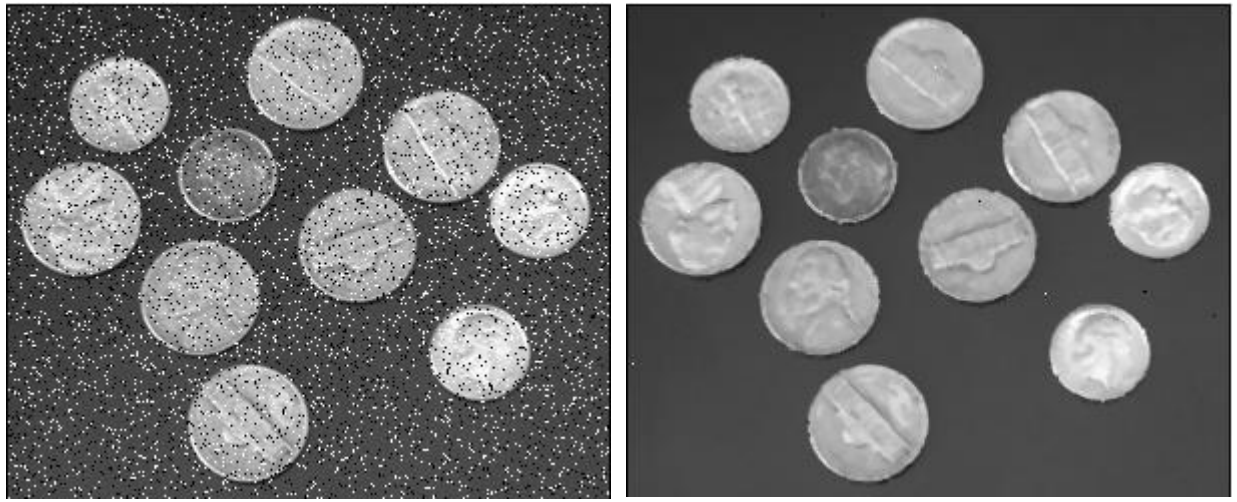
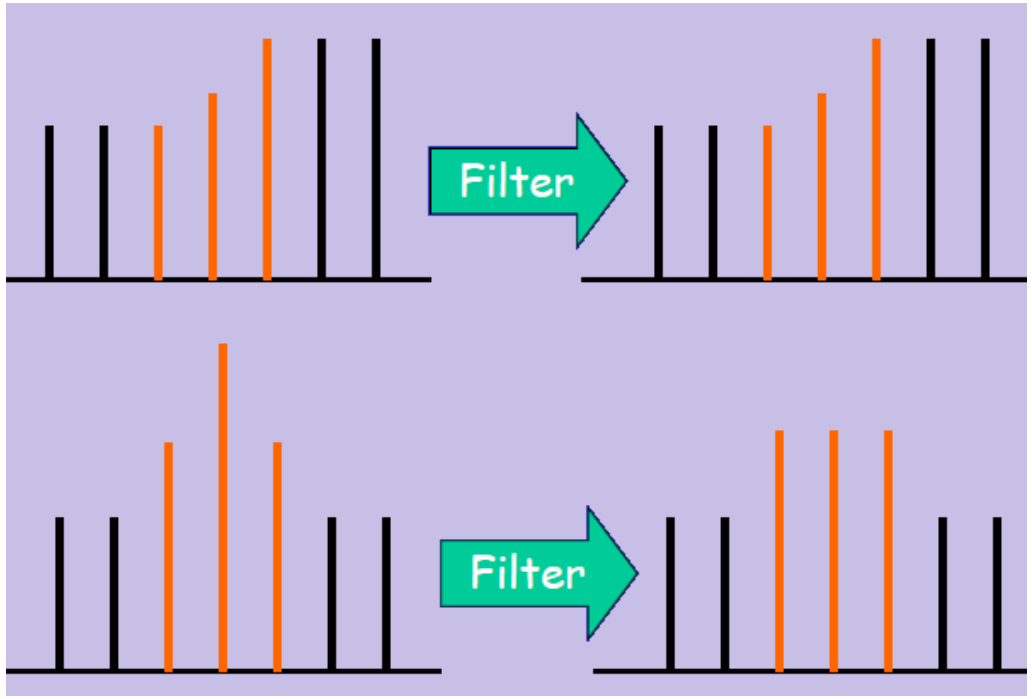


image with massive S&P noise almost perfectly reconstructed by applying a Median filter with r=1



Median filter effect in theory

Relevant Filter Kernels cont'd

- **high-pass** filtering, all exemplarily at radius r=1
  - ***Prewitt*** as rudimentary gradient calculator for horizontal (x) and vertical (y) gradients with

$$K_{prewV} = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad K_{prewH} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

gradient magnitude as $\mathrm{G} = \sqrt{{G_x}^2 + {G_y}^2}$ and edge direction as $\theta = \arctan\left({G_y}/{G_x}\right)$.

  - ***Sobel*** filter with implicit rough Gauss smoothing for increased detection stability as improvement of Prewitt

$$K_{sobelV} = G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad K_{sobelH} = G_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

  - higher accuracy, if also diagonal kernels are incorporated, e.g.

$$K_{prewV1} = G_{xy} = \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$ and so on.

Relevant Filter Kernels cont'd

- **high-pass** filtering, all exemplarily at radius r=1
  - **Robinson operator** (*compass masks*) – rotated by 45° there are 8 directed gradients, namely N, NW, W, SW,…

$$K_{robN} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix} \quad K_{robE} = \begin{bmatrix} -1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & 1 & 1 \end{bmatrix}$$

  - **Kirsch operator**: eight kernels (rotated by 1 position, i.e. 45° each) with 5 times 3 and 3 times -5 sum up to 0.0. The result is defined as maximum filter response applying all 8 possible Kirsch masks as $I'[x,y] = \max_{z=1..8} \sum_{i=-1}^{1} \sum_{j=-1}^{i} K_z[i,j] \cdot I[x+i, y+j]$, thus to be denoted as *non-linear* filter:

$$K_{kirschG1} = \begin{bmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ -5 & -5 & -5 \end{bmatrix} \quad K_{kirschG2} = \begin{bmatrix} 3 & 3 & 3 \\ 3 & 0 & -5 \\ 3 & -5 & -5 \end{bmatrix}$$

Relevant Filter Kernels cont'd

- edge detector results, negative values truncated

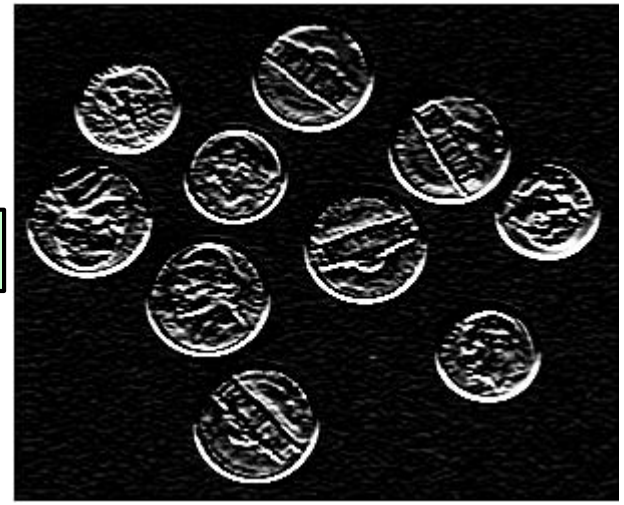
Prewitt horizontal

Prewitt vertical

Sobel horizontal

Sobel vertical

Kirsch G1

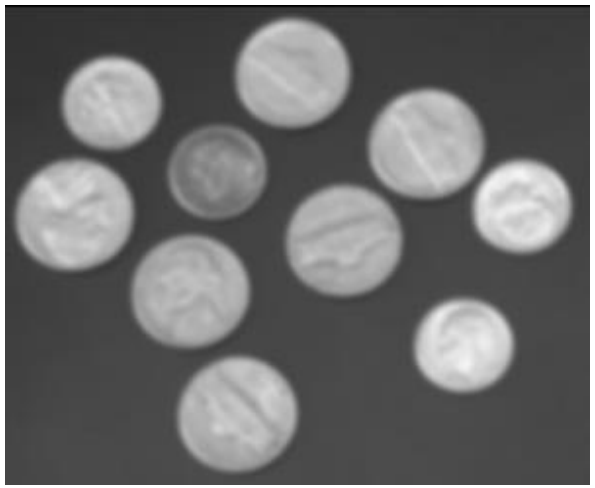


Robinson N,
image inverted
(object expected to be darker)

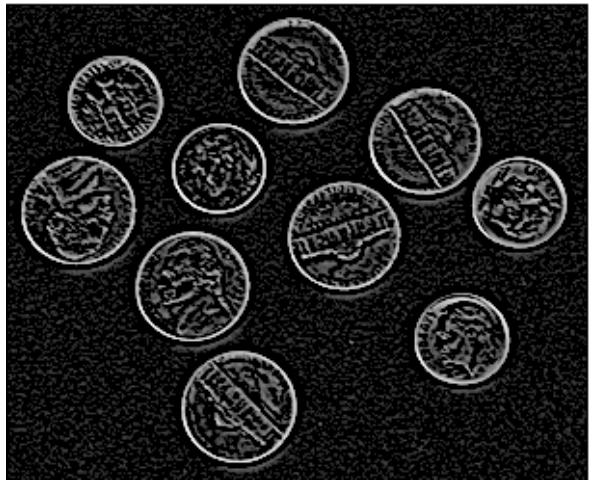
Relevant Filter Kernels cont'd

- edge detectors via low pass-filtering?
  - smoothing primarily affects the edges, thus an edge image can be calculated as $I_{edge} = I - (I \circ K_{gauss})$

I...orig image



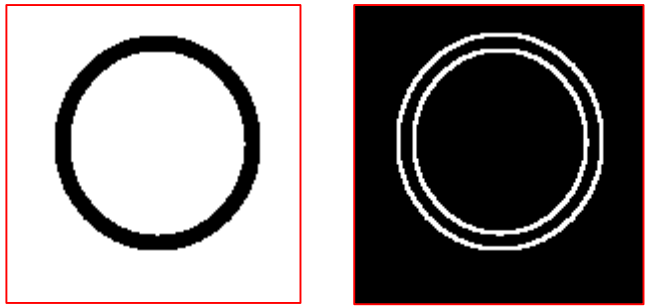
$I \circ K_{gauss}$...smoothing, e.g. Gauss r=3



$I - (I \circ K_{gauss})$

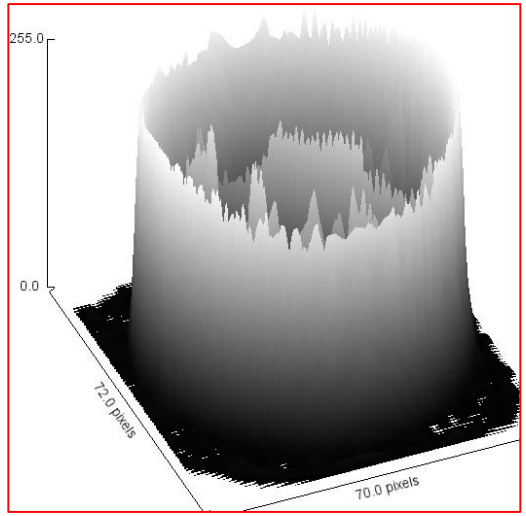
Relevant Filter Kernels cont'd

- response of common edge detectors (Sobel, Prewitt,...) in not normalized scalar range, e.g. [-1020;1020] for common Sobel-mask applied on 8bit image data
- thick lines will lead to inside/outside border response in case of low filter mask radii



thick lines lead to inside and outside gradient response

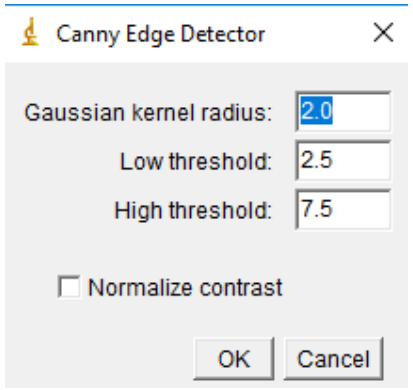
gradients of various thickness and magnitude, c.f. surface plot

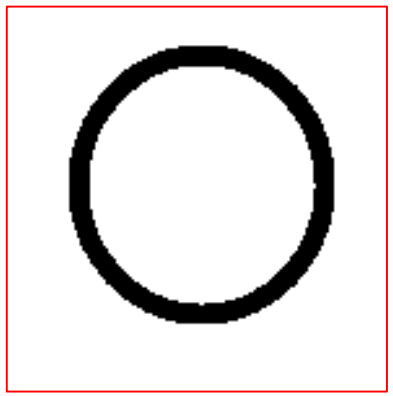


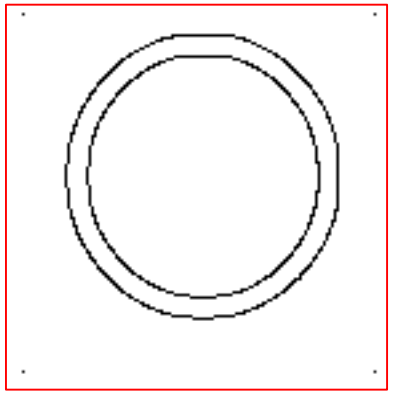
- thus, demand for post-processed edge response is given: **Canny Edge Detector**

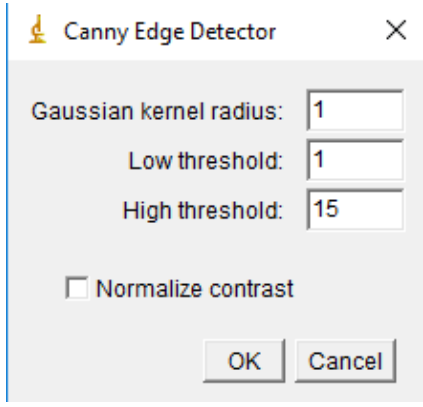
Relevant Filter Kernels cont'd

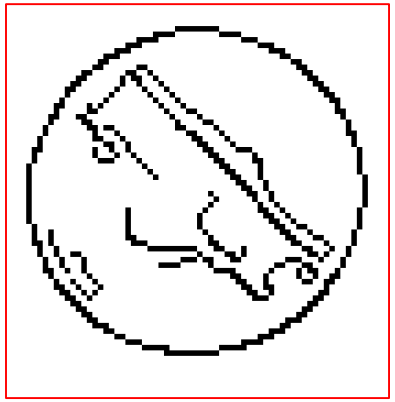
- **Canny Edge Detector** [Canny, 1986]
  - post-processing of Sobel edge detection results and Gauss-pre-filtering
  - first, all local minima of gradient response are detected w.r.t edge direction (lower left and right neighbour of the gradient flank are removed)
  - two threshold values T_1 and T_2 are defined with $T_1 < T_2$. All gradients below T_1 are removed
  - based on edge seed candidates $\geq T_2$, hysteresis is applied, i.e. following the edge direction at the path of highest gradients (all of course $\geq T_1$). The highest value in local neighbourhood is added to extend the edge path
  - final result: **connected** and **binary** path of edges at **constant thickness of 1 pixel**











Canny results with inverted intensities

Relevant Filter Kernels cont'd

- line detectors
  - sum of kernel coefficients =0
  - designed according to target line width and orientation, e.g.:

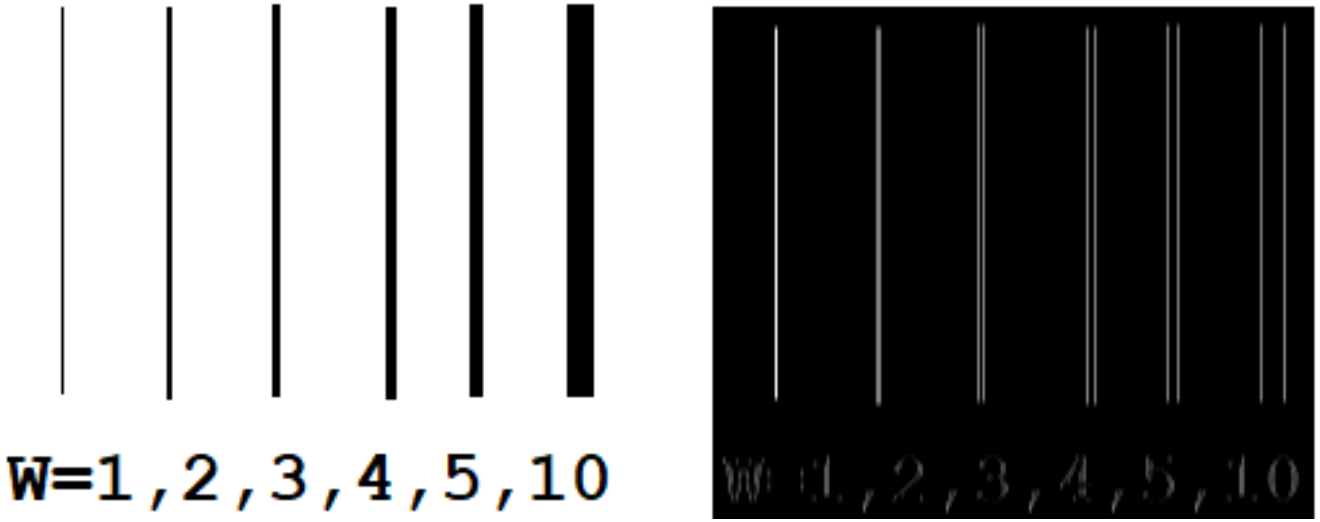
$$K_{lineH1} = \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} \quad K_{lineV1} = \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix} \quad K_{lineDiag1} = \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix}$$

  - in case of applying e.g. K_{lineH1} on edge areas or thicker lines, around half the filter response compared to "matching" lines is observed
  - How to detect horizontal lines of width $w = 3$?
    - maybe design a kerned with $r = 2$, e.g.
    - or: [*better*] keep K_{lineH1} and **scale the image**!
    - by scaling the image, detection of arbitrary line width besides regular image grid becomes feasible, e.g. $width = 2.2$

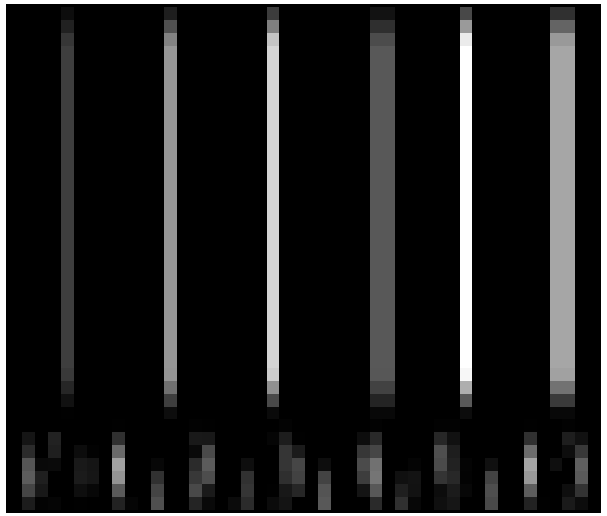
$$K_{lineV3} = \begin{bmatrix} -3 & 2 & 2 & 2 & -3 \\ -3 & 2 & 2 & 2 & -3 \\ -3 & 2 & 2 & 2 & -3 \\ -3 & 2 & 2 & 2 & -3 \\ -3 & 2 & 2 & 2 & -3 \end{bmatrix}$$

Relevant Filter Kernels cont'd

- line detection results:



detection of bars at varying width. in case of perfect match, a solid filter response is observed, while thicker lines lead to double-line results showing lower scalar magnitude



reducing scale to 0.2, the highest response is observed for bar of width=5 The quality of results thereby depends on the *interpolation strategy*

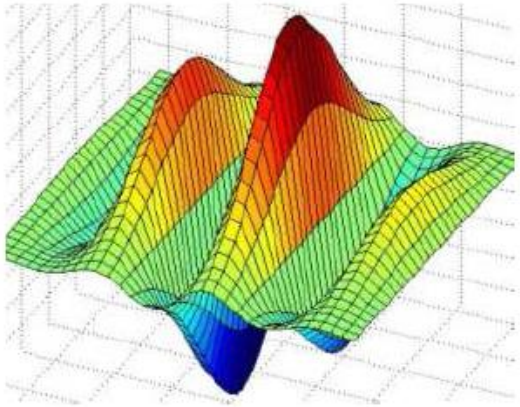
Relevant Filter Kernels cont'd

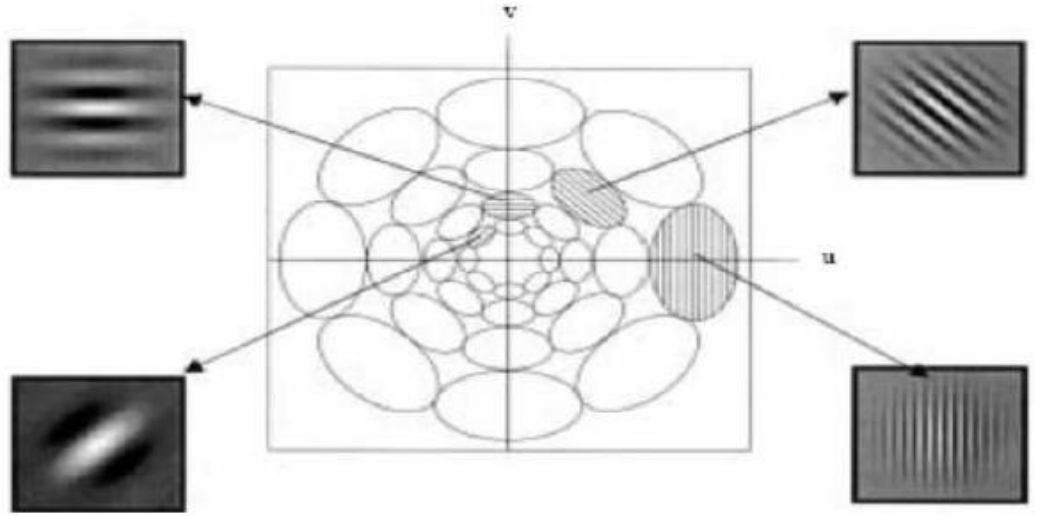
- relevant filter kernels: **high-pass detectors**
  - **point detector** (minimum, maximum)

$$K_{point} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

  - complex filters such as **Gabor kernel** with $r >> 1$ apply to discrete filter mask too (same convolution algorithm!).
    - with θ for structure orientation, phase-shift ψ, wave-length λ, aspect-ratio γ and σ for sinusoidal smoothing effect of Gabor filter.

$$g(x, y, \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right)\exp\left(i\left(2\pi\frac{x'}{\lambda} + \psi\right)\right)$$





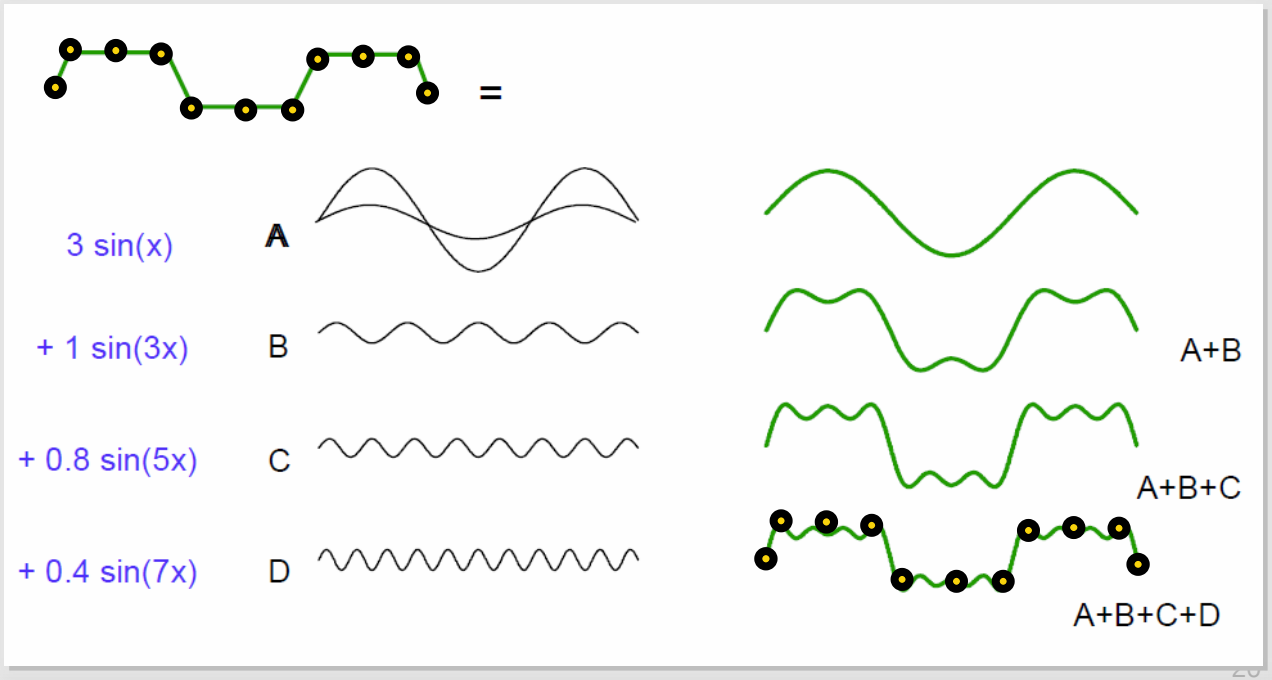
Frequency Domain Filtering

- *Jean Baptiste Joseph Fourier* (1768-1830)
  - French physicist and mathematician
  - inventor of Fourier Transformation
    - no additional information is obtained
    - given information is transformed into frequency dom
    - transformation is reversible without loss (loss-less)
    - key idea/principle:

all arbitrary functions can be represented as sum of weighted cos and sin waves of varying frequency and amplitude

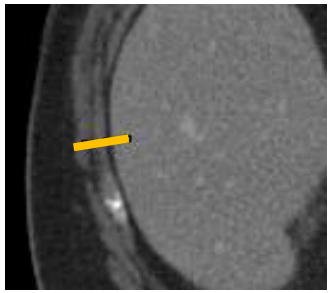
Frequency Domain Filtering cont'd

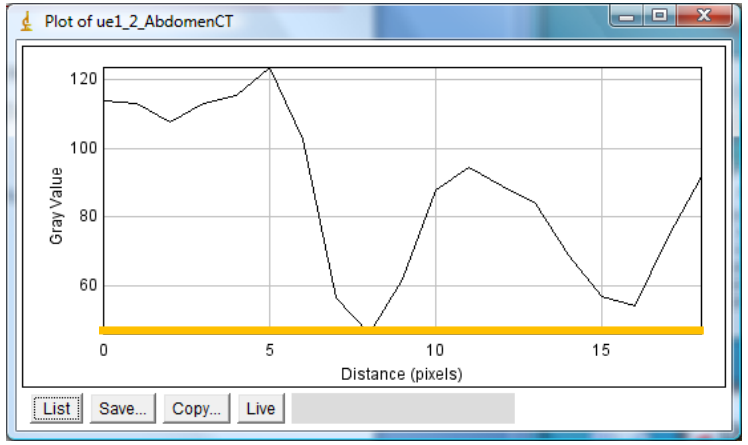
- 1D Fourier Transformation
  - an arbitrary discrete signal can be composed by a sum of sufficient ***sin*** and ***cos*** functions with varying frequency and amplitude
  - an continuous signal can only be reconstructed from an infinite number of sin/cos frequencies
  - discrete: e.g. transformation of a rectangular-shaped signal
  - an image is interpreted as columns/rows of arbitrary signals and thus Fourier Transformation is applicable too
  - But is that a sufficient approximation of the perfect rectangular shape?
    - considering only the relevant discrete positions: **YES!**

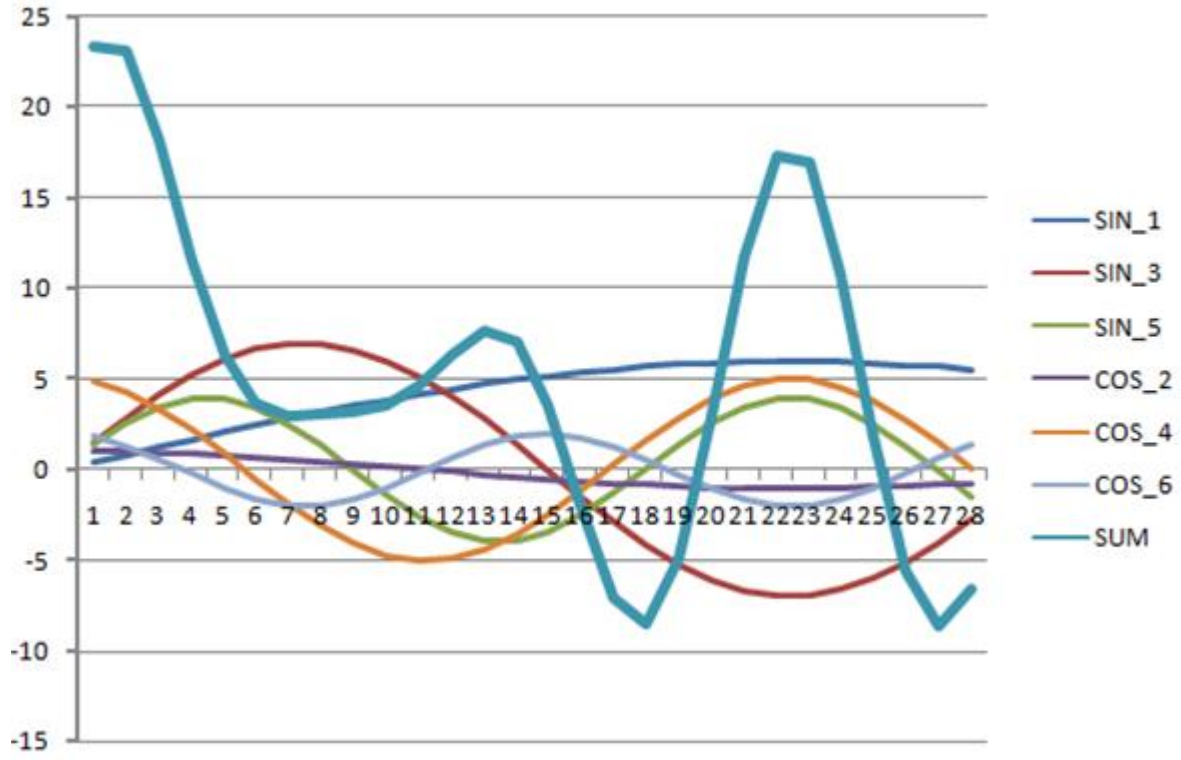


Frequency Domain Filtering cont'd

- but where do we have these 1D signals in images?
  - line profile showing arbitrary shape
  - this shape can be approximated by a sufficient number of sin/cos functions with adequate frequency and amplitude

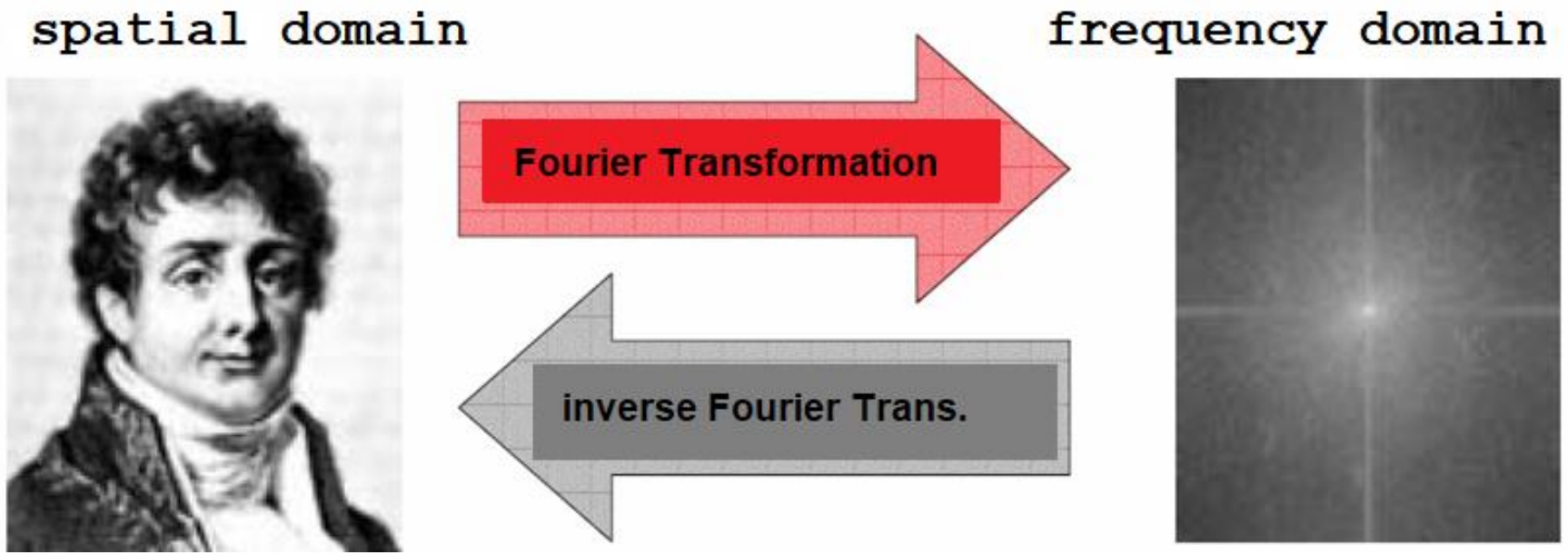






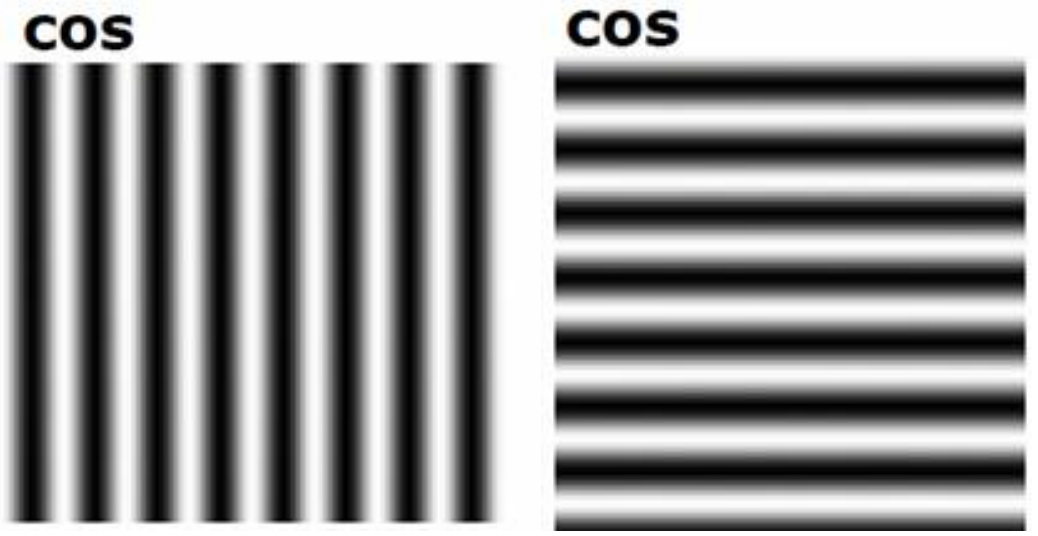
Frequency Domain Filtering cont'd

- principles of Fourier Analysis:
  - in the ***spatial domain*** an image is represented by scalar values assigned to discrete pixel coordinates (x, y)
  - in the frequency domain, an image is represented by scalar amplitudes assigned to discrete positions (u, v) representing a particular sin/cos wave
  - *an image, i.e. a 2D signal, can be transformed from spatial domain to frequency domain and vice versa in an unambiguous and lossless way*



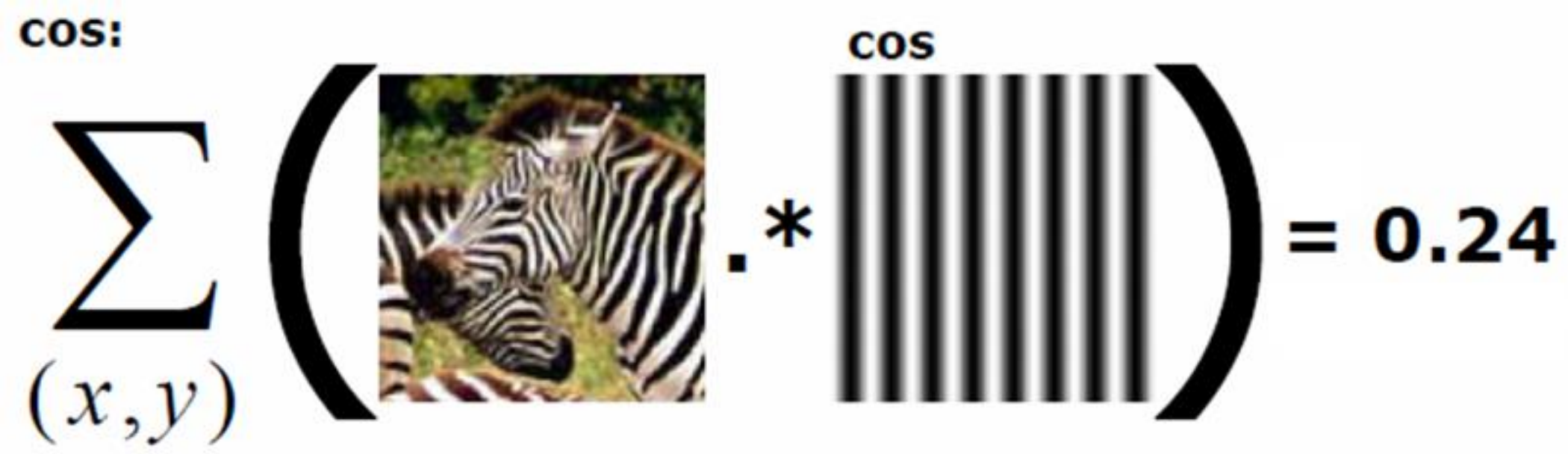
Frequency Domain Filtering cont'd

- How are the amplitudes and frequencies derived?
  - periodic sin/cos patterns of varying orientation and frequency are pre-defined in the spatial domain according to (u, v)-ratio and radius
  - the "correct" amplitude for each of the periodic patterns is calculated by pointwise multiplication
  - exaggerated illustration: which pattern is of higher importance in the image below? The vertical or the horizontal cos-pattern?



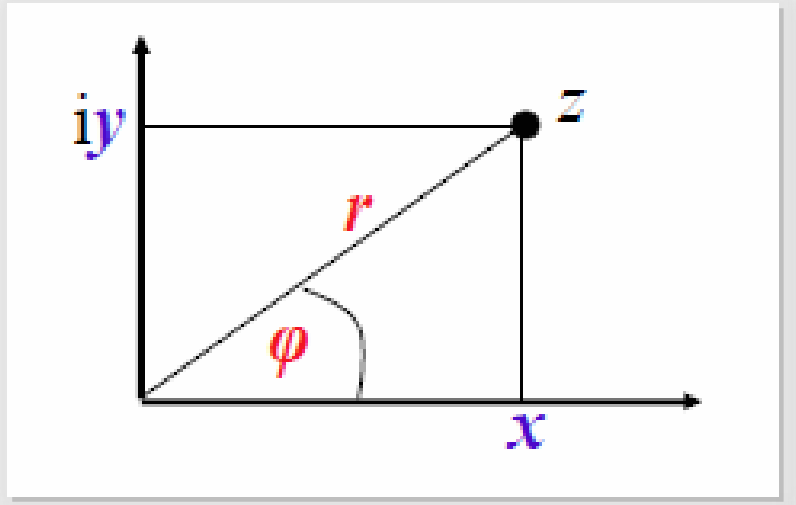
Frequency Domain Filtering cont'd

- How are the amplitudes and frequencies derived?
  - for each of the sin/cos patterns, the input image is pointwise multiplied (.*), summing up the products (∑) for particular amplitude value
  - the higher the amplitude, the more important the particular wave-pattern is for image reconstruction



Frequency Domain Filtering cont'd

- sin/cos represented by polar coordinates with angle φ representing the orientation of the waves in the pattern
  - $z = \exp(i\varphi) = \cos(\varphi) + i \cdot sin(\varphi)$ with $i = \sqrt{-1}$
- for complex number z, a representation in both, *polar* and *Cartesian* space exists:
  - Cartesian: $z = (x + i \cdot y)$
  - polar: $z = \exp(i\varphi) = \cos(\varphi) + i \cdot sin(\varphi) = r \cdot \exp(i\varphi)$
    - with amplitude $r = \sqrt{x^2 + y^2}$ and
    - amplitude $\varphi = atan2(y, x)$



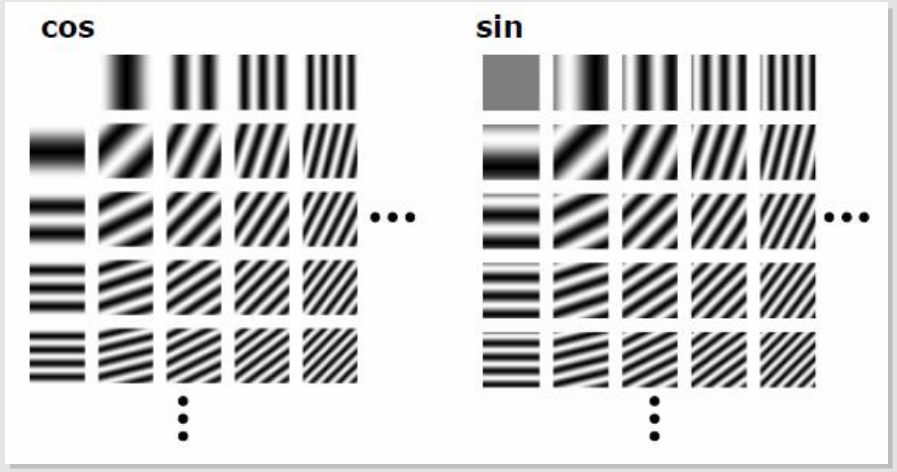
url link, bold, [8pt]

Frequency Domain Filtering cont'd

- formula to transform image **g** with size *M,N* from spatial domain (x, y) to frequency domain (u, v):
  - with $u = 0,1, \ldots, M-1$ and $v = 0,1, \ldots, N-1$ as coordinates of **G** and x, y coordinates of g

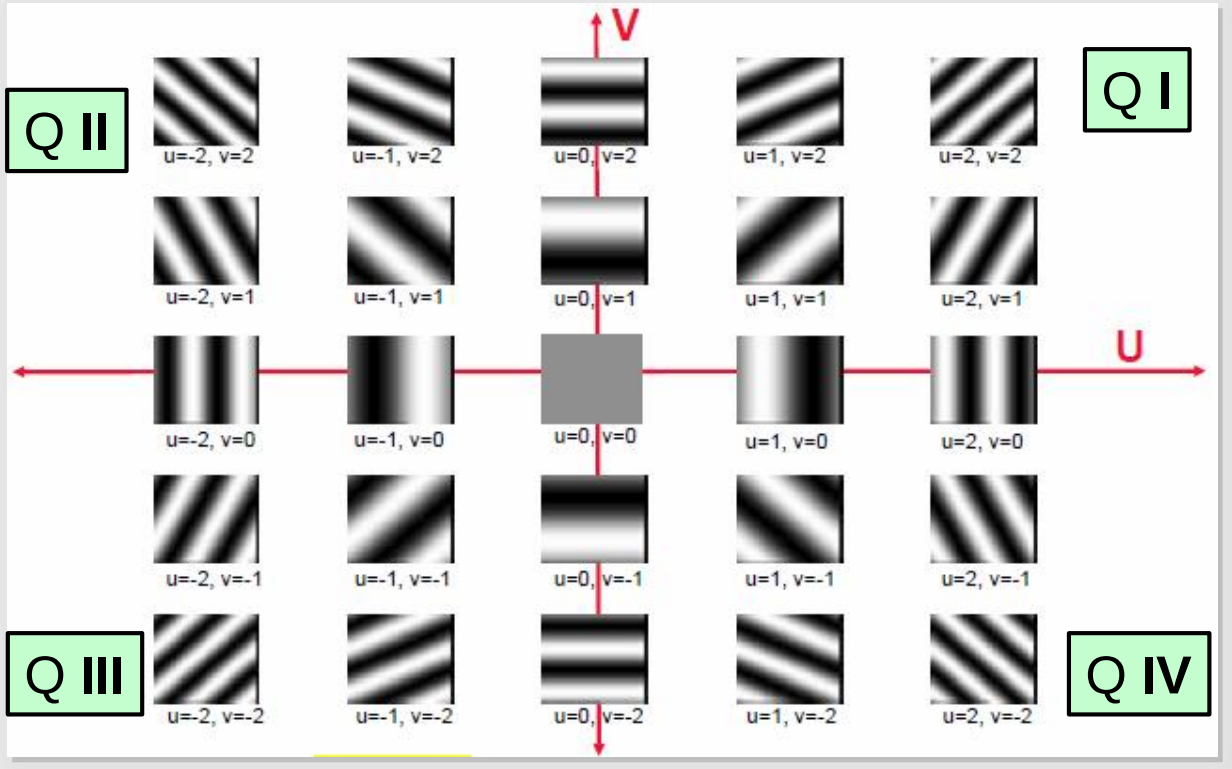
$$G(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} g(x,y) \exp\left(-i2\pi \left(\frac{ux}{M} + \frac{vy}{N}\right)\right)$$

  - iterating that way, the low frequencies (main information) are located in the top left corner while the higher frequencies (fine structures, details) are located "outside"



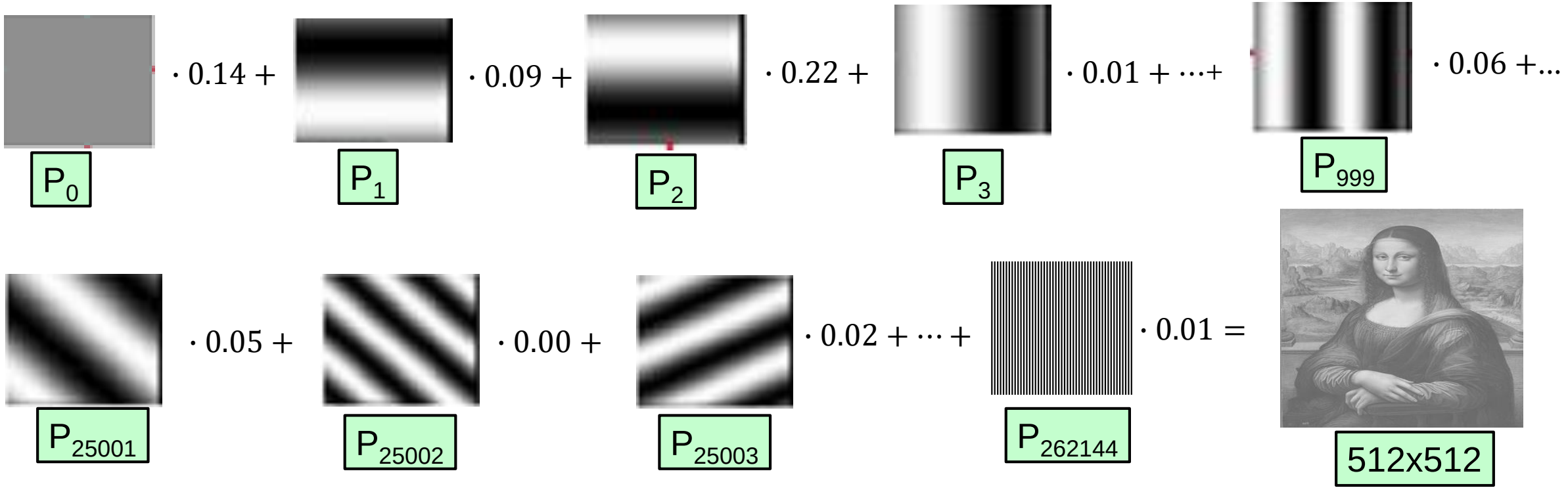
Frequency Domain Filtering cont'd

- for input image of size $N \times M$, there are $2 \times N \times M$ sin/cos patterns projected onto the original image
- representation with low frequencies in the top left corner as calculated hard to interpret
  - thus, FFT-shift is applied leading to a redundant representation with lowest frequencies in the centre and high frequencies outside
  - ratio u/v thereby determines the orientation and wave-length is calculated by $\frac{1}{\sqrt{u^2+v^2}}$
  - quadrants **I** and **III** as well as **II** and **IV** respectively are holding the same information (*redundancy*)



Frequency Domain Filtering cont'd

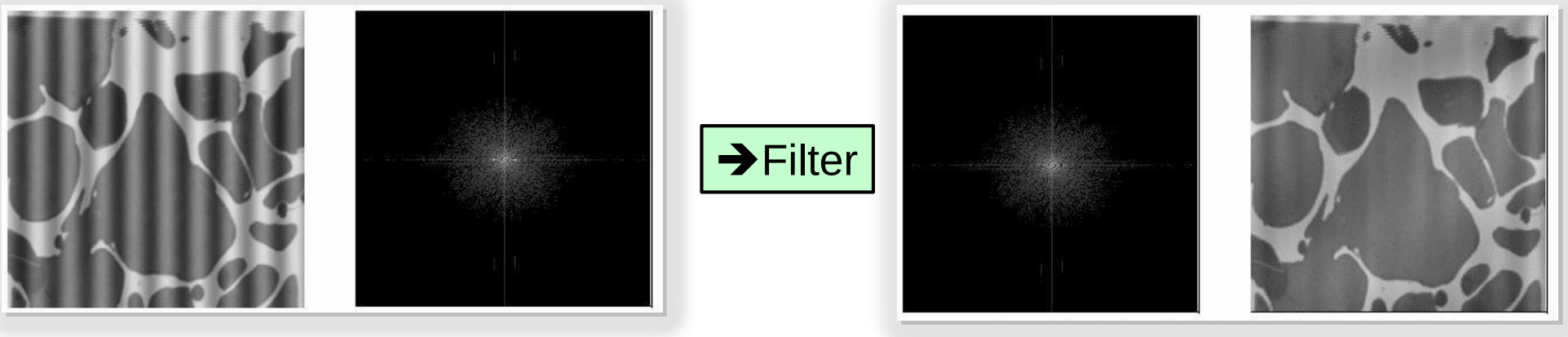
- recapitulation for a better understanding
  - using a huge number of different sin/cos images multiplied by the correct amplitude and summing up all these matrix-products the original image can be reconstructed as inverse Fourier Transformation



Frequency Domain Filtering cont'd

- images in the frequency domain:
  - spectrum of natural images quite the same

  - in case of noise or artificial images, the spectrum will significantly deviate



Frequency Domain Filtering cont'd

- Filtering in the Frequency Domain:
  - while in the spatial domain convolution needs to be applied, in the frequency domain the Fourier-transformed image and kernel can be multiplied (thus need to be of matching size)
- Fourier Transformation from spatial domain to frequency domain:
  - $g(x) \rightarrow G(\varphi)$
- in the frequency domain, filtering is performed as multiplication by kernel K
  - $G' = G \times K$
- finally, the image needs to be transformed back into spatial domain
  - $G'(\varphi) \rightarrow g'(x)$
  - for back-transformation (IFT) almost the same formula as sued for Fourier-Transformation (FT) is applicable, besides normalization part and imaginary-part ***i*** with negative sign:

$$g(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} G(u,v) \exp\left(i 2\pi \left(\frac{ux}{M} + \frac{vy}{N}\right)\right)$$

Frequency Domain Filtering cont'd

- convolution theorem:
  - the convolution of two functions ***f*** and ***g*** in the spatial domain can be achieved by multiplication of the Fourier-transformed ***F*** and ***G*** in the frequency domain:

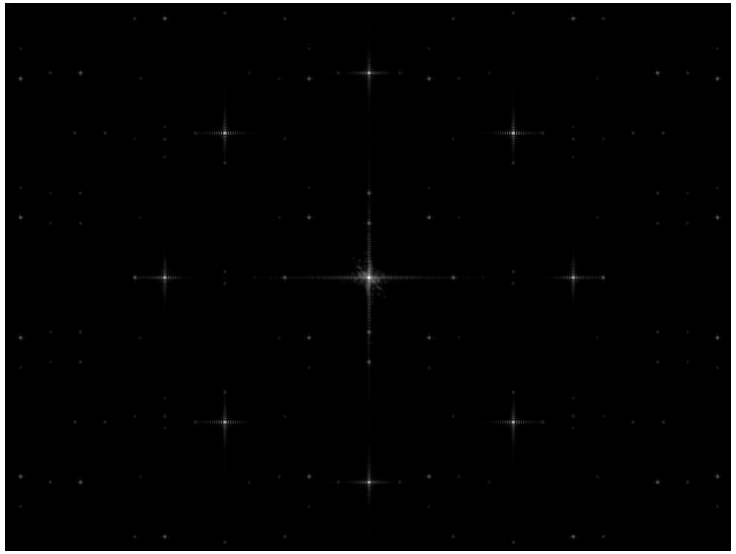
$$- f(x) \circ g(x) \equiv F(\varphi) \times G(\varphi)$$

- Fast Fourier Transformation (FFT) applicable if the image dimensions are a multiplicate of 2, e.g. 512, 1024, aso.
  - in case of processing images of lower dimension, the dimensions must be enlarged w.r.t. FFT calculation premise
  - various concepts are applied for speed-up
  - iterative transformation of multi-dimensional input signals
  - generally, only the magnitude is considered/visualized with $magnitude(F) = \sqrt{real(F)^2 + imag(F)^2}$

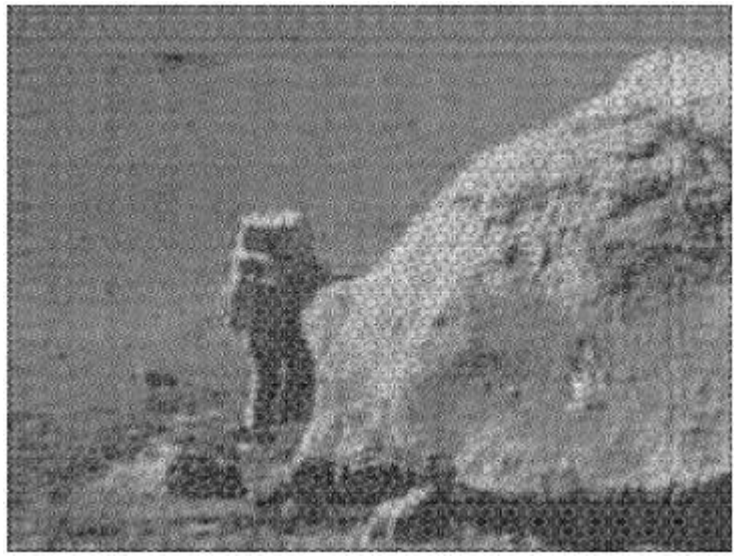
Frequency Domain Filtering cont'd

- Filtering in the Frequency Domain:
  - removal of periodic noise, c.f. [http://stackoverflow.com/questions/5165587/removing-sinusoidal-noise-with-butterworth-filter]

image **I**



FT(**I**)



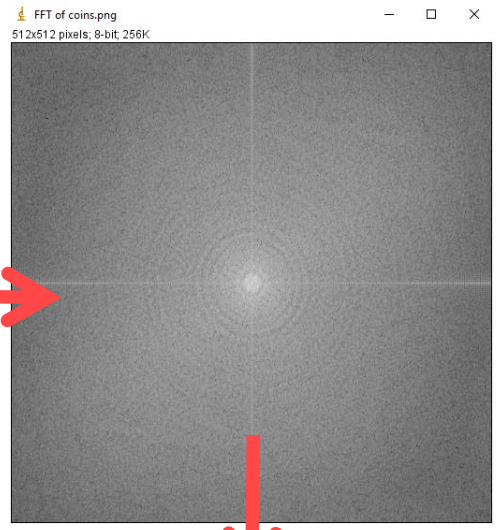
IFT(*FT*(**I**)**FT*(**K**))

Frequency Domain Filtering cont'd

- high- and low-pass filtering by amplifying/damping the high and low frequencies respectively

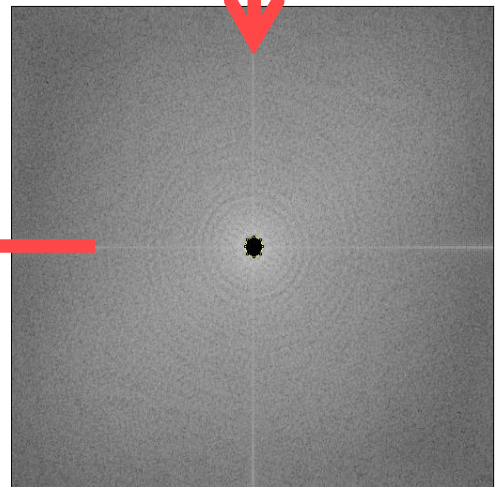
input image, size 300x246

FFT



FFT of input image, size 512x512

erase ~1% of the lowest-frequency amplitudes



IFFT

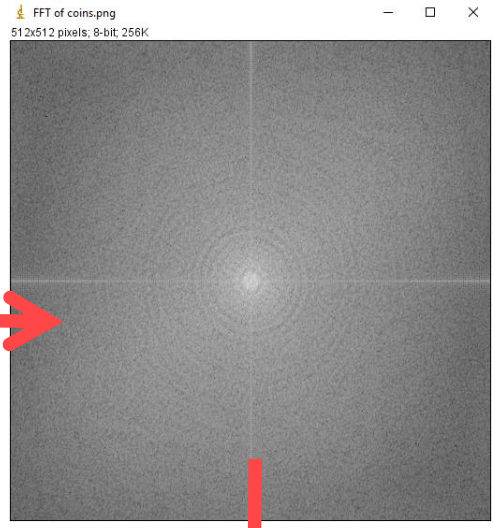
resulting **edge detetion**

Frequency Domain Filtering cont'd

- high- and low-pass filtering by amplifying/damping the high and low frequencies respectively

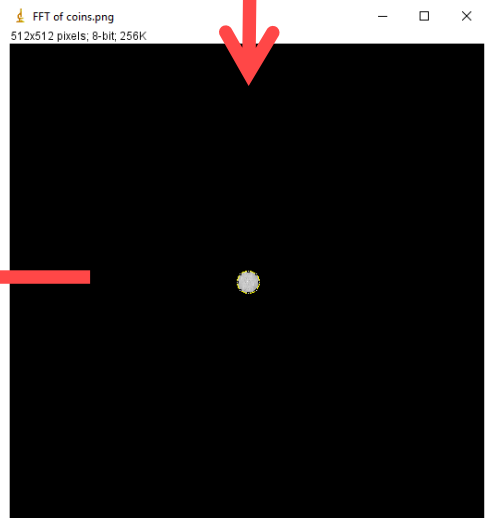
input image, size 300x246

FFT



FFT of input image, size 512x512

erase ~99% of the highest-frequency amplitudes

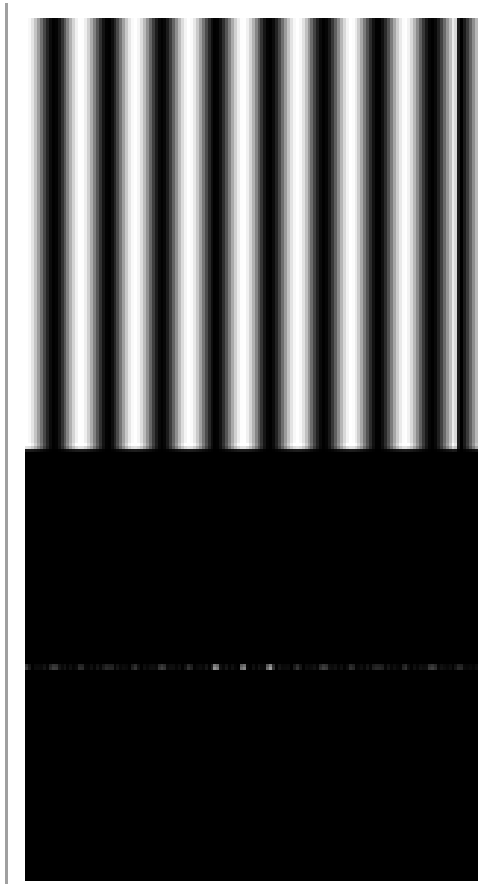


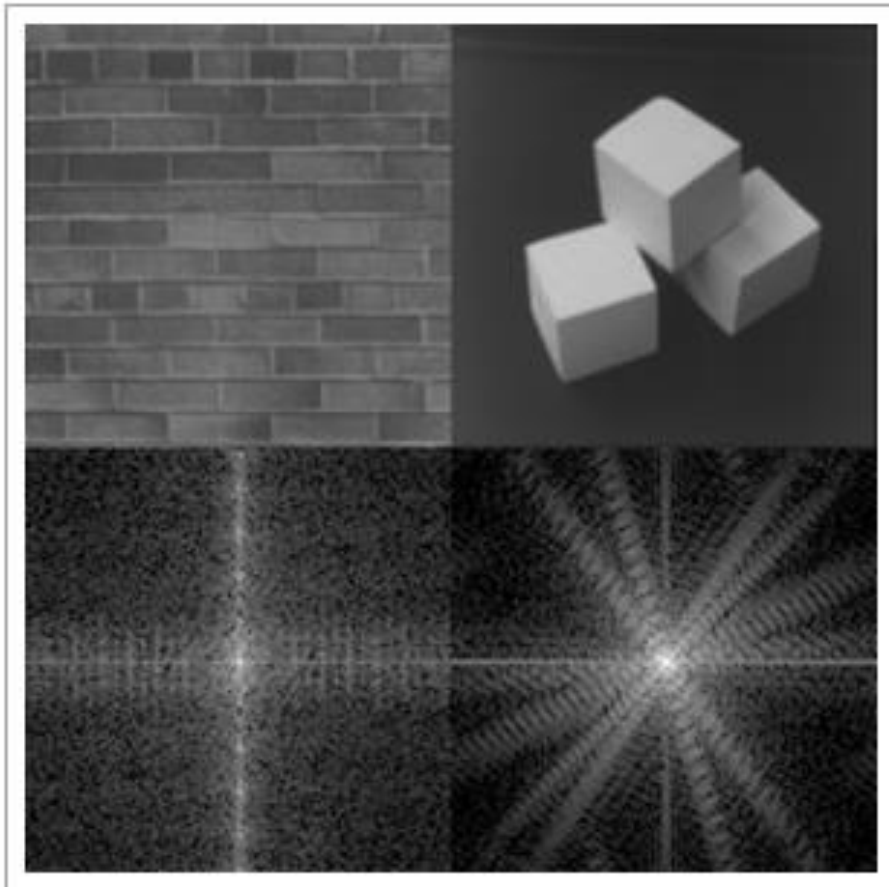
IFFT

resulting image **smoothed**

Frequency Domain Filtering cont'd

- based on the magnitude of the Fourier-transformed signal, periodic structures can be identified:

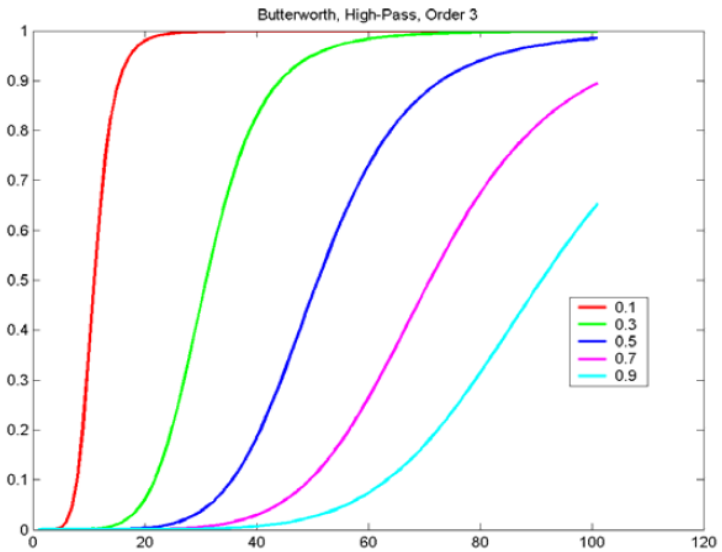


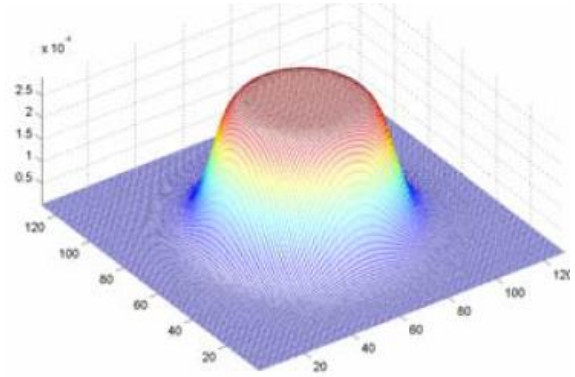


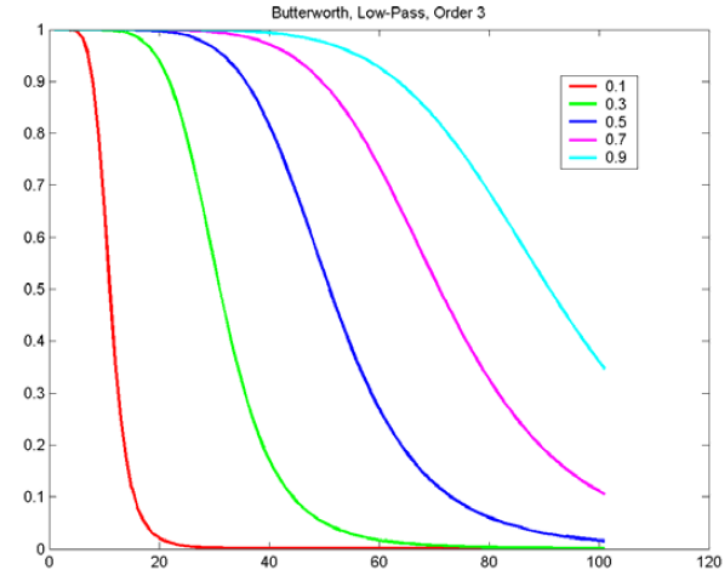
from [http://www.cs.unm.edu/~brayer/vision/fourier.html]

Frequency Domain Filtering cont'd

- filtering in the frequency domain:
  - instead of binary masking amplitudes of particular high or low frequency range, a smooth damping is preferred.
  - Butterworth filter applicable as low- and high-pass filter by parametrizing the cuttof-frequency $\boldsymbol{f_c}$ and exponential order $\boldsymbol{n}$ (improved quality at significantly increased calculation overhead)
  - high-pass: $k(f) = \frac{1}{1+\left(\frac{f}{f_c}\right)^{2n}}$ low-pass: $k(f) = \frac{1}{1+\left(\frac{f_c}{f}\right)^{2n}}$



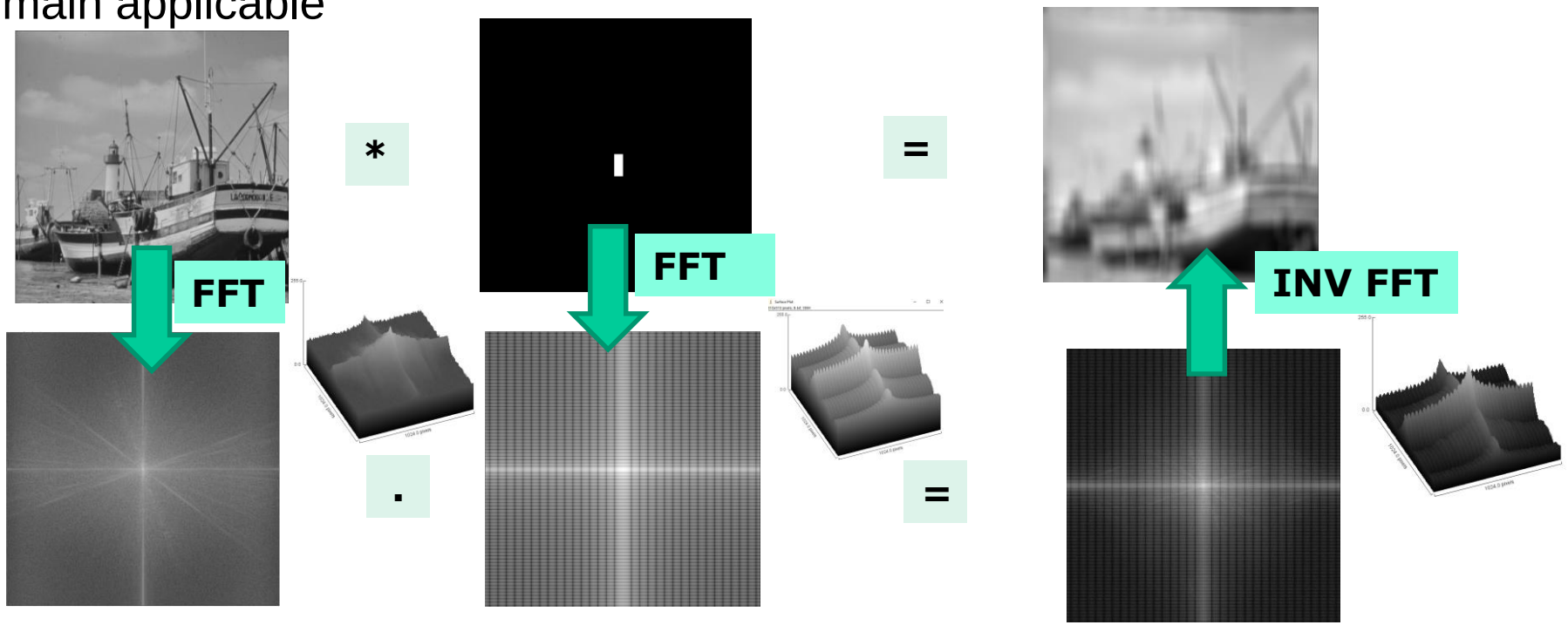




$\boldsymbol{f_c}$ in [0.1;0.9] bei $\boldsymbol{n}$ =3

Frequency Domain Filtering cont'd

- **Outlook**: Adaptive Filtering, Deconvolution
  - knowing the kernel K that affects quality of measured image (e.g. motion during image acquisition), reconstruction becomes feasible dividing by K in case of no noise present
  - $g(x,y) = f(x,y) * h(x,y)$ ➔$G^*(u,v) = \frac{F(u,v)}{H(u,v)}$ but attention: DIV 0 possible
  - **Wiener Filter** in Frequency and **Richardson-Lucy-Deconvolution** [Richardson 1972; Lucy 1974] in Spatial Domain applicable



Bibliography

- J. Canny, “A Computational Approach to Edge Detection”, 1986.
- L.B. Lucy, “An iterative technique for the rectification of observed distributions”. In: *Astronomical Journal*, vol. 79, 1974
- William Hadley Richardson, “Bayesian-Based Iterative Method of Image Reconstruction”. In: *Journal of the Optical Society of America*, vol. 62, issue 1, 1972.

UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

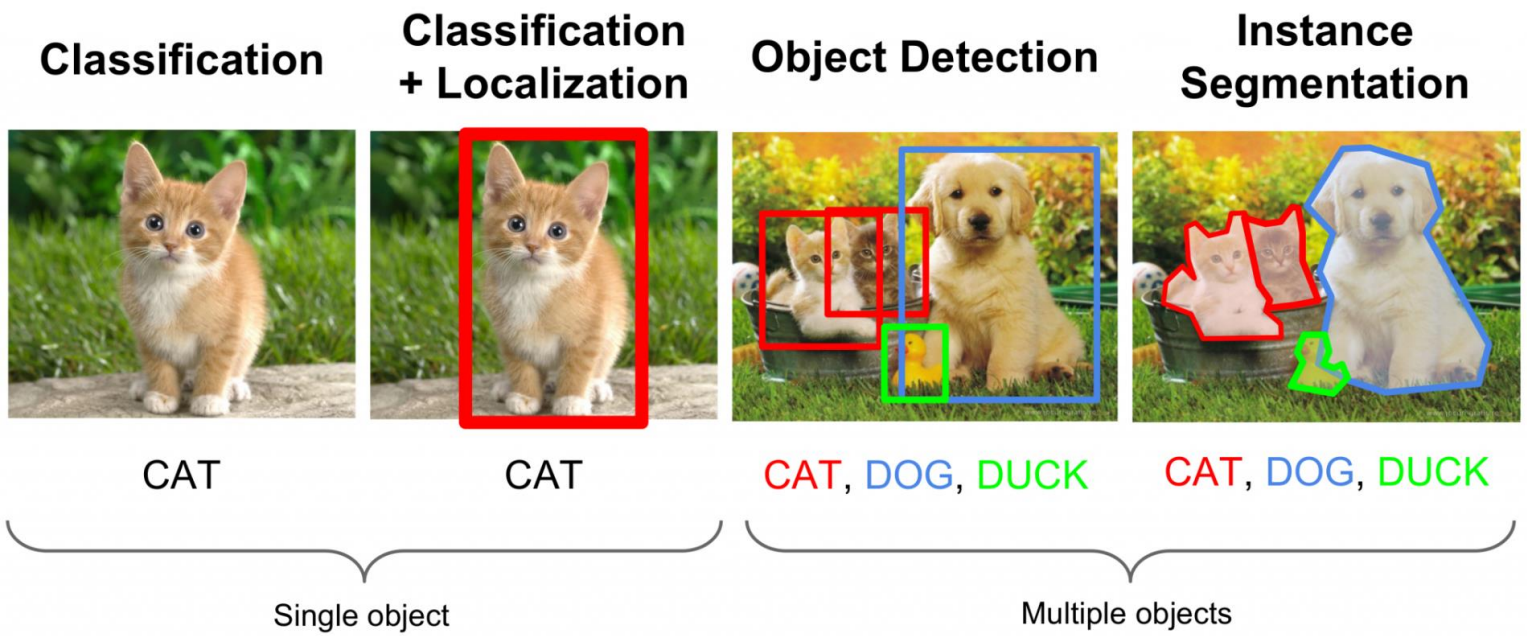
Image Processing

Segmentation

DI(FH) Dr. Gerald Adam Zwettler, MSc, gerald.zwettler@fh-hagenberg.at
Lector of Computer Graphics, Image Processing and Computer Vision

Segmentation Overview

- segmentation refers to voxel-wise classification of the pixels of an input image ***A***.
- segmentation /classification as step towards semantic interpretation of the image
  - tagging of the entire image by one “word” ➔ Classification
  - additionally detecting the ROI around the potentially occluded key object ➔ Object detection with *Classification* and *Localization* for images with 1 or more objects involved
  - marking the precise visible or occluded pixel region referring to the classified object ➔ Instance **segmentation**

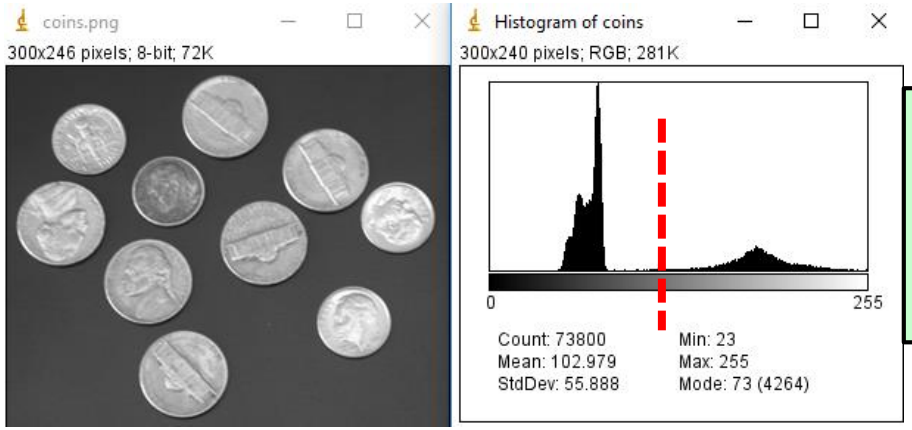


Segmentation Overview cont'd

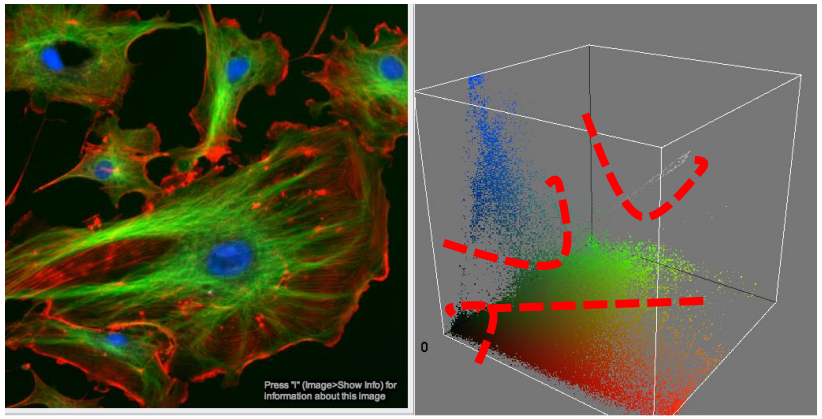
- motivation, fields of application
  - pre-requisite in vision-based monitoring of industrial processes – is the workpiece intact
  - medicine/biology: number of cells in microscopy, 3D reconstruction of patient's anatomy, surgical planning,...
  - forensics: exact measurement on image content
  - general: object detection (characters for OCR, faces,...)

Segmentation Methods

- features and characteristics to detect objects in images:
  - segmentation algorithms exploit the same characteristics that are relevant for human vision, i.e. colour, illumination, shape, gradients, texture,...
  - simplest segmentation approaches exclusively utilize intensities or colours.
    - Object classes to segment/discriminate thereby expected show varying intensity statistics, all often following a normal distribution $\mathcal{F}_j \sim \mathrm{N}(\mu, \sigma)$
    - Feasibility of colour or intensity based segmentation can be deduced from histogram analysis: if the feature statistics for two classes $\mathrm{X}_1, \mathrm{X}_2$ are well separated and hardly overlapping, an intensity-based segmentation approach might be sufficient.



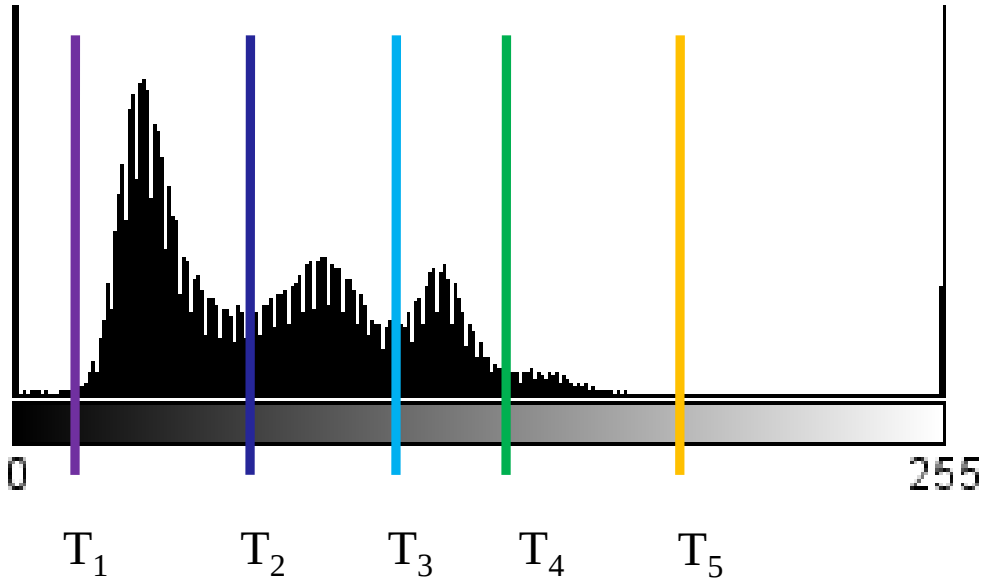
coins and background clearly separated with $\mu_{coin} \approx 180$ and $\mu_{bg} \approx 85$

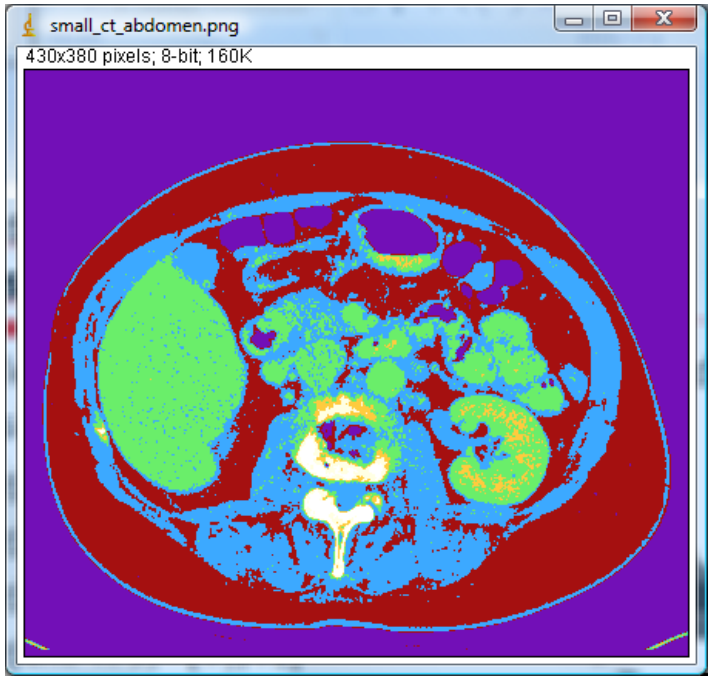


3D histogram, ColorInspector

Segmentation Methods cont'd

- Intensity based Thresholding (*Schwellenwert*)
  - to discriminate ***n*** classes, a total of ***n-1*** thresholds per channel (1 for grayvalue intensities, 3 in case of RGB) are defined to classify each pixel.

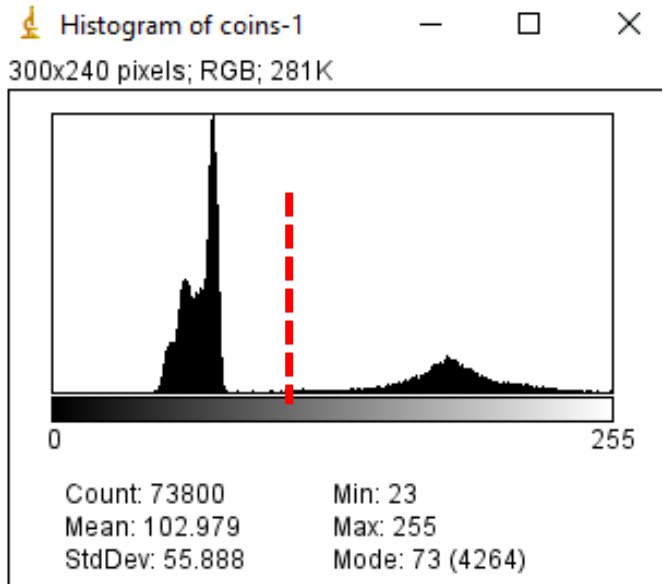


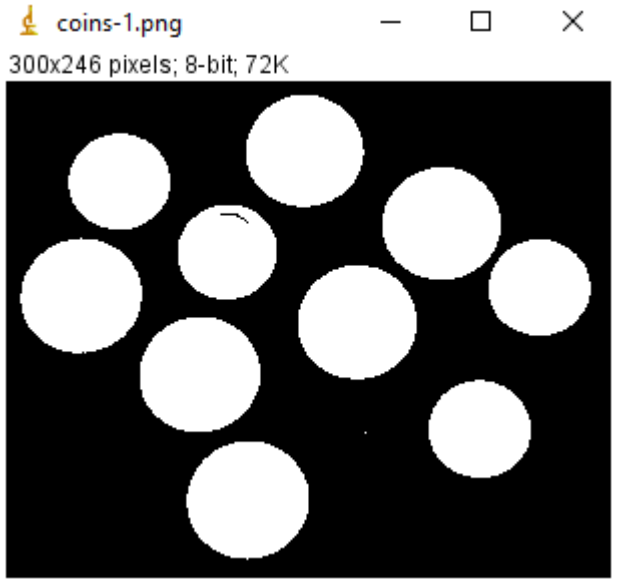


  - threshold-based segmentation: each pixel is uniquely assigned a class according to scalar value

Segmentation Methods cont'd

- Optimal Threshold
  - in case of bi-modal histogram, the threshold value **T** best discriminating the objects left and right of **T** can be calculated in an automatic way, the so called *optimal threshold*.



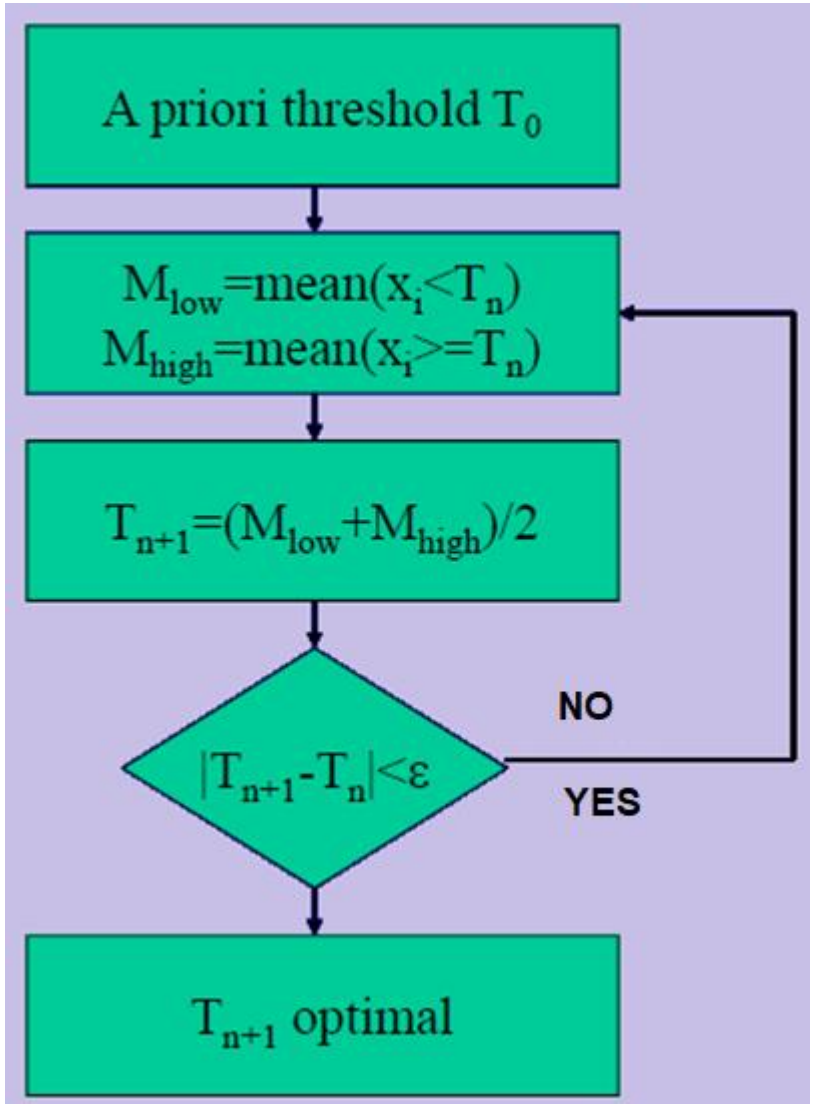


url link, bold, [8pt]

T=102 as final result

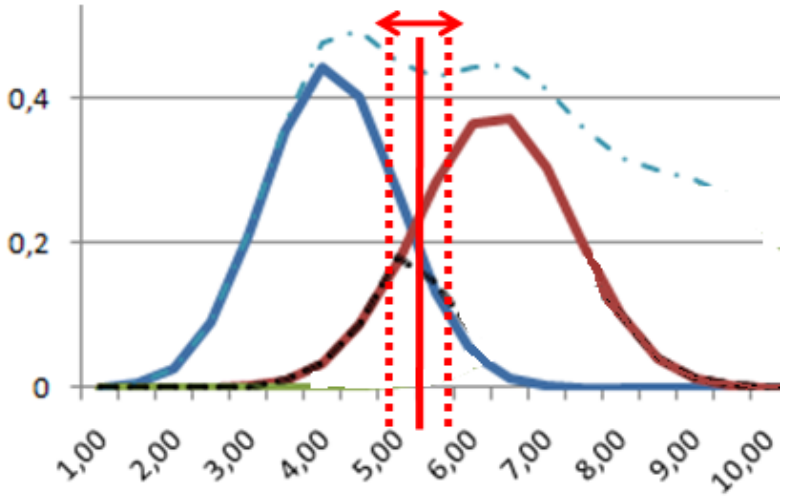
Segmentation Methods cont'd

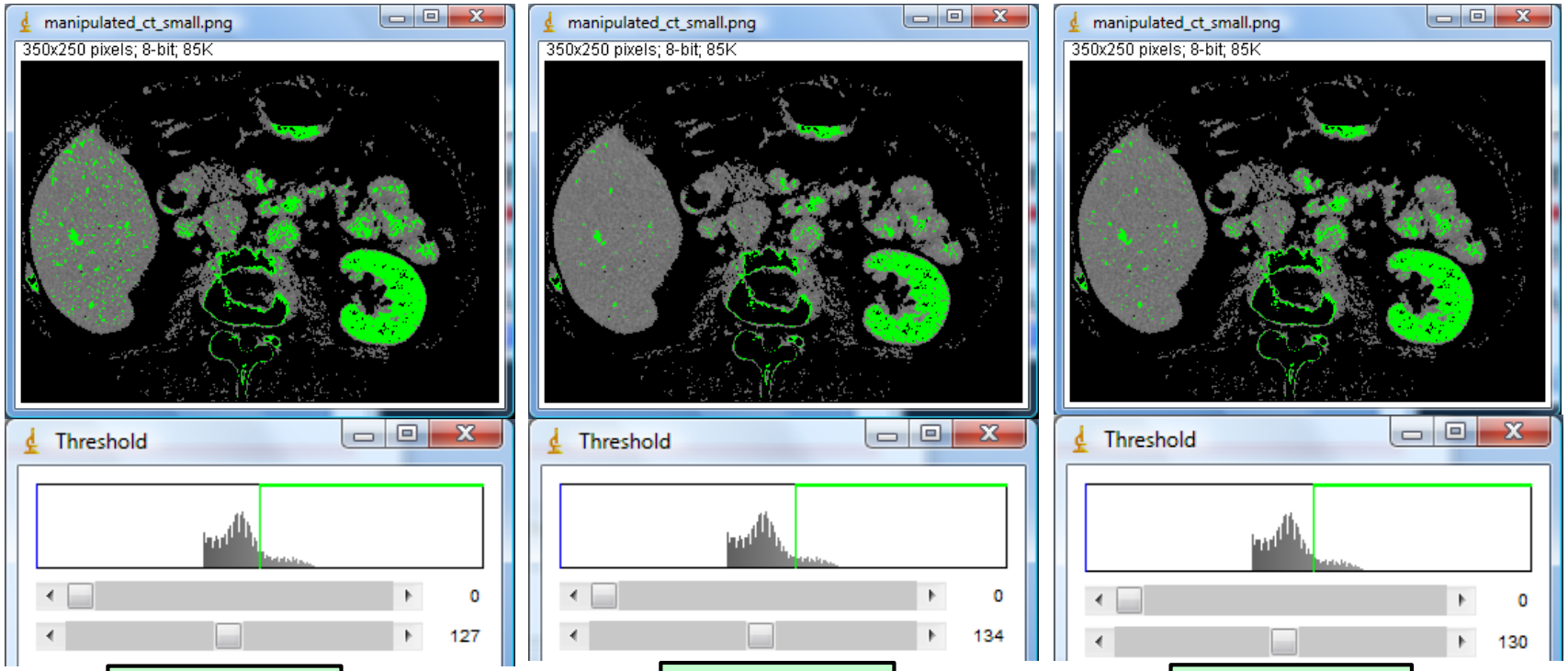
- Optimal Threshold cont'd
  - in case of overlapping distributions of the classes X_1, X_2, the optimal T is iteratively approximated
  - thereby the Bayes error below the cumulated distributions of X_1, X_2 is minimized
    - starting with an initial guess T_0, e.g. the mid of scalar range (127 for 8bit)
    - (a) mean of values below and above current threshold are calculated with $\mu_{low1} = mean(x_i < T)$ and $\mu_{high1} = mean(x_i \geq T)$
    - (b) update T as $T_{n+1} = \frac{\mu_{low\_n} + \mu_{high\_n}}{2}$
    - repeat (a) until convergence is reached, e.g. $|T_{n+1} - T_n| < \varepsilon$. T_{n+1} then considered the *optimal threshold*



Segmentation Methods cont'd

- Optimal Threshold cont'd
  - the thresholds T_n are optimized until convergence is reached





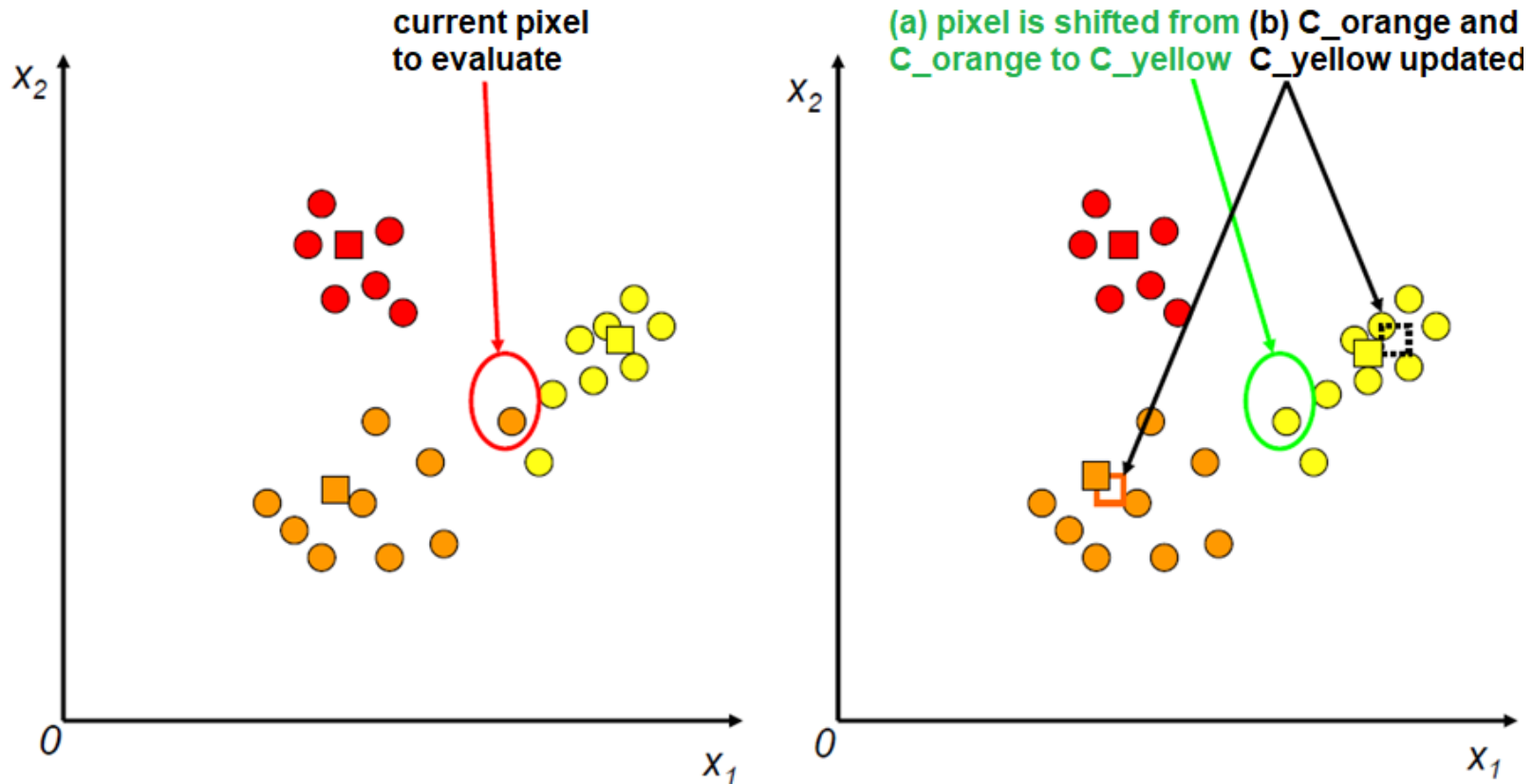
Iteration 1 | Iteration 2 | Iteration N

Segmentation Methods cont'd

- KMeans Clustering
  - while Optimal Threshold is applicable to bi-modal distributions only, KMeans-Clustering allows processing of *n-modal* distributions
  - pixels are assigned the clusters according to minimal distance, e.g. colours in 3D colour space
  - number of clusters and the initial cluster positions are pre-defined. Number of clusters thereby pre-determines the segmentation granularity
  - during optimization, empty cluster assignments to be considered
  - KMeans Clustering as optimization procedure:
    - all pixels are initially assigned the cluster showing the lowest distance
    - then, the cluster centroid is updated according to the assigned pixel values
    - pixels are checked again w.r.t. closest cluster – if the assignment is changed, the clusters need to be updated again
    - KMeans clustering stops if convergence is reached, i.e. the pixels are robustly assigned

Segmentation Methods cont'd

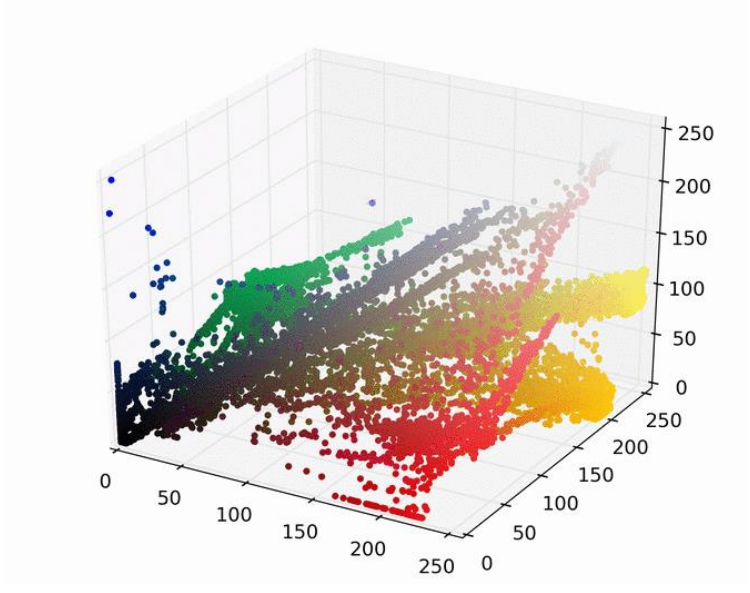
- KMeans Clustering cont'd
  - optimization process illustrated for 2D colour space. The colour clusters (rectangular) are re-calculated from all pixel colours (circle) that are assigned according to proximity.
  - after updating the cluster, the assignments might need to be updated aso.



Segmentation Methods cont'd

- KMeans Clustering cont'd
  - if color space shows dominant clusters, KMeans clustering or similar approaches such as Mean Shift will lead to sufficient pre-segmentations

initial colour image

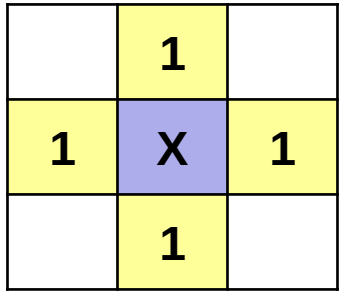


3D colour space showing local clusters. Gray-values generally outspread due to illumination variance

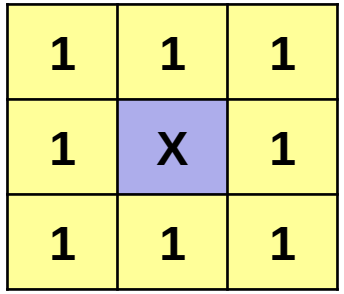
result: clusters get assigned the final centroide mean

Segmentation Methods cont'd

- Region Growing
  - whenever a global intensity threshold or colour cluster is not feasible due to several objects showing similar characteristics, *region growing* might be applicable if the structures hard to discriminate are at least locally separable
  - besides the intensity interval, local neighbourhood adjacency is claimed.
  - the user thereby "selects" the target region by setting a seed marker
  - all pixels showing matching intensities that are connected to the seed marker are part of the result segmentation.



N_4



N_8

Segmentation Methods cont'd

- Region Growing cont'd
  - starting at the seed, all neighbouring pixels fulfilling the intensity requirements are added and propagated in a recursive way
  - example with interval [2;4]

	1	
1	X	1
	1	

N_4

input image

0	1	3	3	4	6
0	1	2	2	2	5
1	2	2	4	3	7
0	1	1	4	5	6
0	3	6	8	9	7
1	2	3	6	5	6
3	3	6	2	2	1

result with plain interval threshold. The 2-3 yellow regions cannot be separated

seed

iteration #1

iteration #2

iteration #n

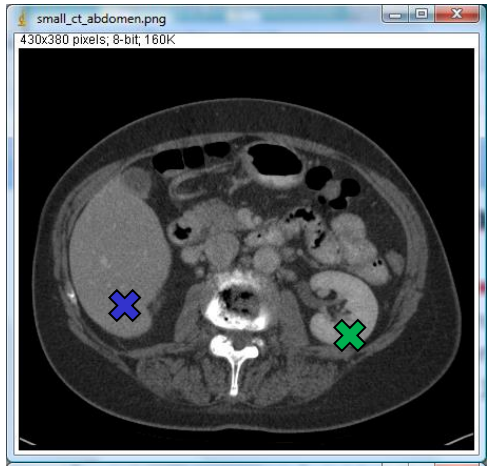
Segmentation Methods cont'd

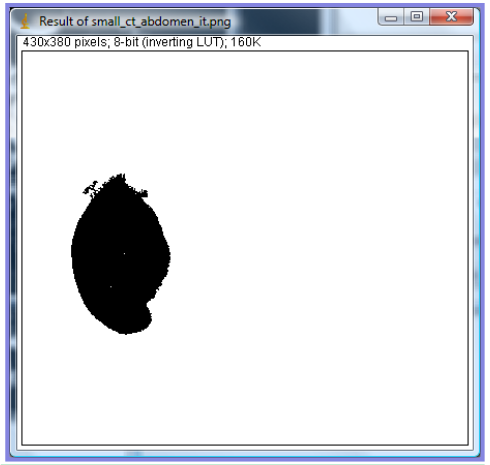
- Region Growing cont'd
  - recursive implementation as Pseudo Code utilizing N_4 with ***I***... input image, ***B***... result image, ***x/y*** current pixel position and ***Tmin,Tmax*** as threshold interval:

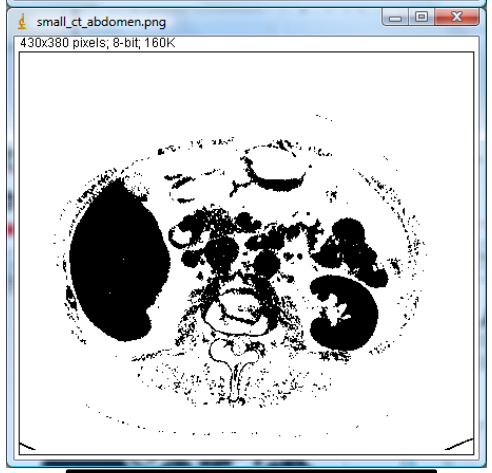
```
grow(I, B, x, y, Tmin, Tmax) {
B(x,y) = 1 //mark pixel as foreground
if B(x+1,y) !=1  && I(x+1,y) >= Tmin && I(x+1,y)<= Tmax
   grow(I, B, x+1, y, Tmin, Tmax)
if B(x-1,y) != 1  && I(x-1,y) >= Tmin && I(x-1,y) <= Tmax
   grow(I, B, x-1, y, Tmin, Tmax)
if B(x,y+1) != 1  && I(x,y+1) >= Tmin && I(x,y+1) <= Tmax
   grow(I, B, x, y+1, Tmin, Tmax)
if B(x,y-1) != 1  && I(x,y-1) >= Tmin && I(x,y-1) <= Tmax
   grow(I, B, x, y-1, Tmin, Tmax)
}
```

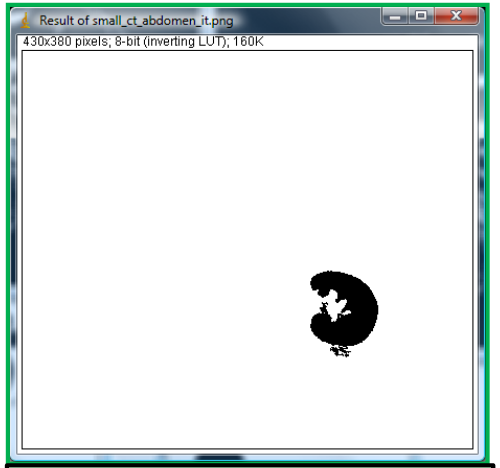
Segmentation Methods cont'd

- Region Growing examples:





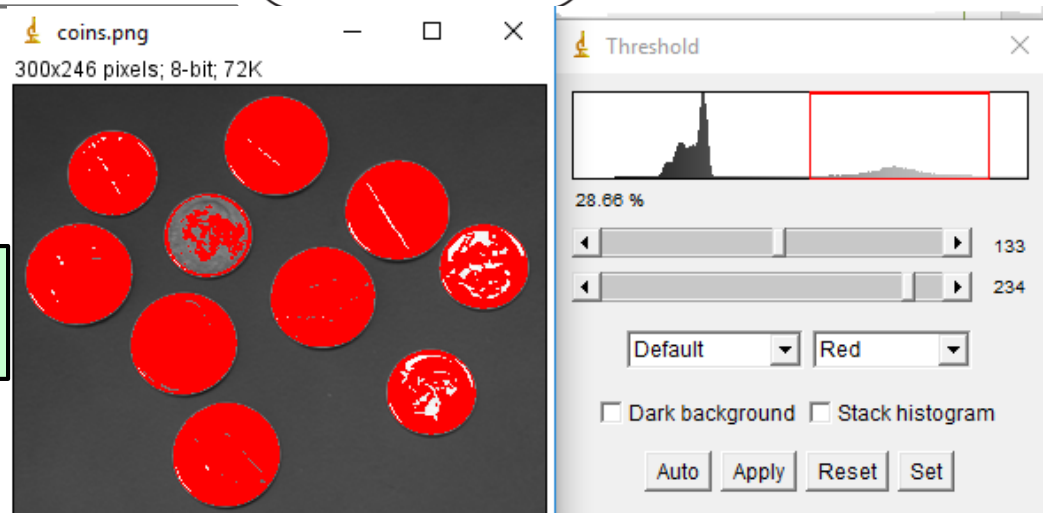


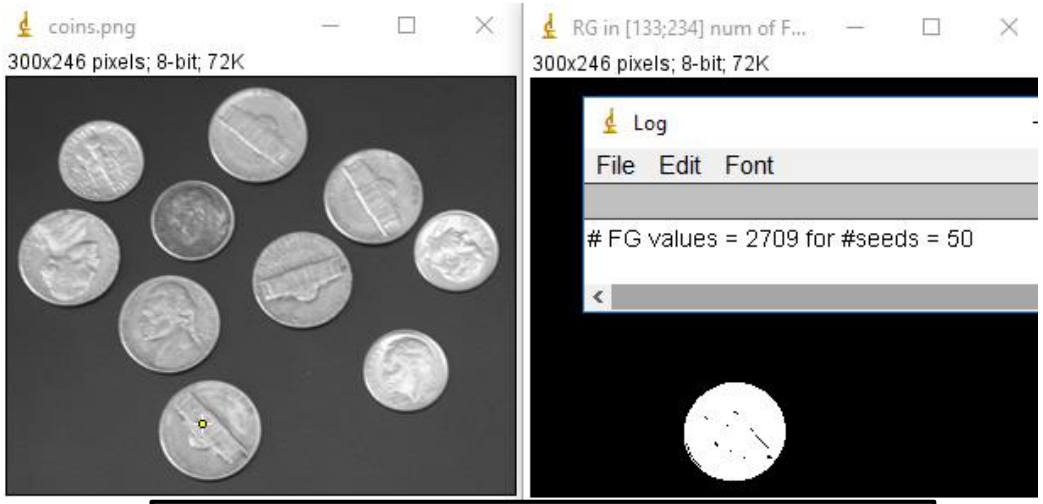


interval thresh with [94;173]

segmentations according to seed

interval thresh with [133;234]

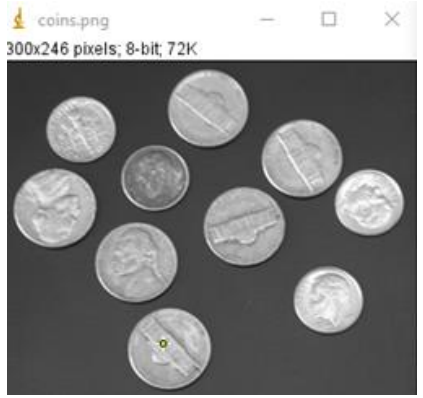




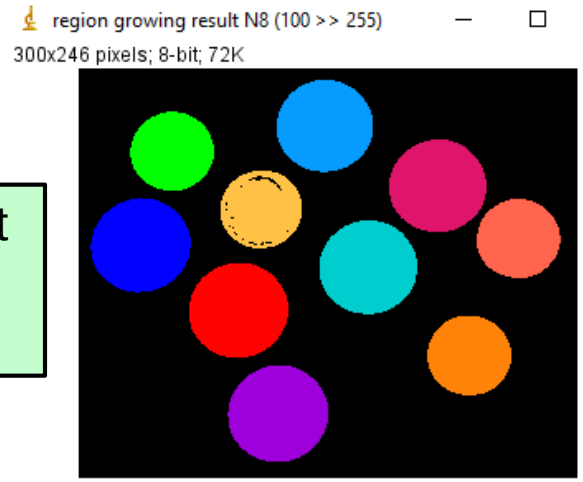
only coin with seed is segmented

Segmentation Methods cont'd

- Variations of Region Growing
  - *confidence connected thresholding*
    - besides the seed position(s), no additional user input is required
    - the interval threshold is calculated based on the seed pixel value $I(x_s, y_s)$ and some pre-defined confidence σ calculating the interval threshold as
      $$[I(x_s, y_s) - \sigma \cdot (I_{max} - I_{min}); I(x_s, y_s) + \sigma \cdot (I_{max} - I_{min})]$$
  - *region labelling*
    - no seeds are provided besides an interval threshold.
    - thus, all interval-threshold segmented pixels are implicit seed categories for region growing assigning an incrementing ID for all of the regions



all connected regions showing target intensity interval [133;234] are assigned an unique ID



Segmentation Methods cont'd

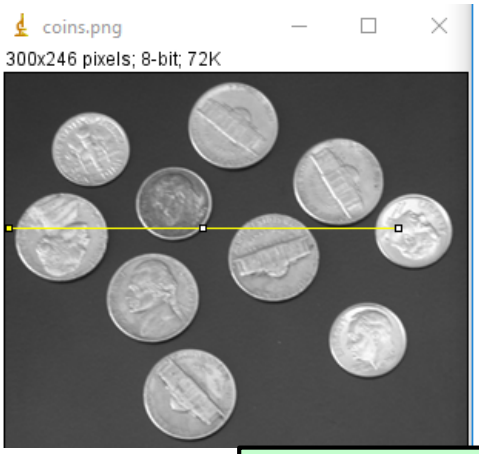
- besides intensities, **gradients** enclosing the separable objects, are another significant characteristics for segmentation approaches
- **watershed transformation**:
  - a border image I_b (grayscale) is derived from input image *I*.
  - objects expected to be surrounded by borders and showing homogenous intensity/chromaticity/luminance.
  - a position detected by a minimum filer applied on I_b (c.f. median or convolution masks) is a good starting candidate for an object region to grow within the borders.
  - processing on [0;255] intensity range, a minimum filter would lead to an immense number of object seed candidates in case of common background noise level. Thus, massive over-fragmentation would occur hardening the later to follow feature-based classification approach.
    - thus, threshold T is introduced as quantization, removing local noise-affected value deviation from the group of potential minima.

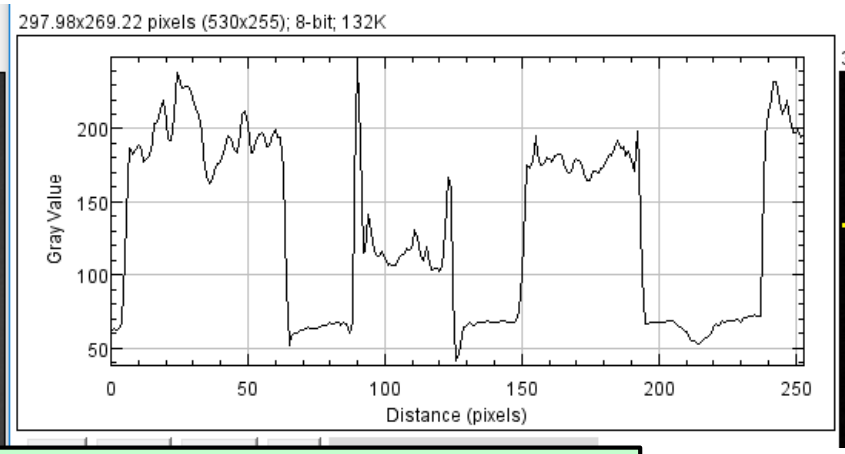
Segmentation Methods cont'd

- **watershed transformation cont'd**:
  - the watershed transform interprets the border image as topography that is equally flooded with the increasing water to flow towards the minima positions.
    - with increased water level one has to define the consequence of two growing neighboring regions finally colliding at the toughing adjacent border area
    - a parameter L for target watershed granularity level defines height of the watershed, i.e. separation barrier between neighboring regions. If the water even so overcomes the barriers, the two affected regions are considered belonging to the same object and thus are merged steering the fragmentation granularity.

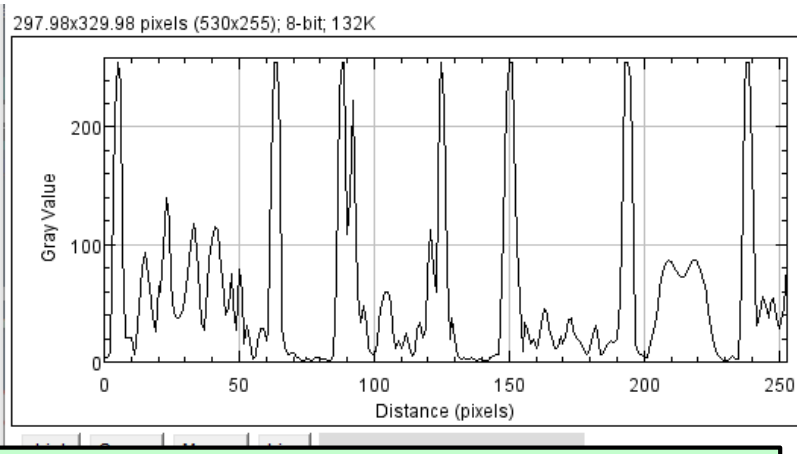
Segmentation Methods cont'd

- **watershed transformation cont'd**:
  - process steps:
    - local minima detection on border image I_b quantized by threshold parameter t as $t(I_b)$
    - iterative flooding, i.e. increasing intensity difference tolerance on adjoined unclassified pixels to be merged with the current basin
    - merge of colliding adjoined regions, if flood level overcomes the watershed level l as separation barrier





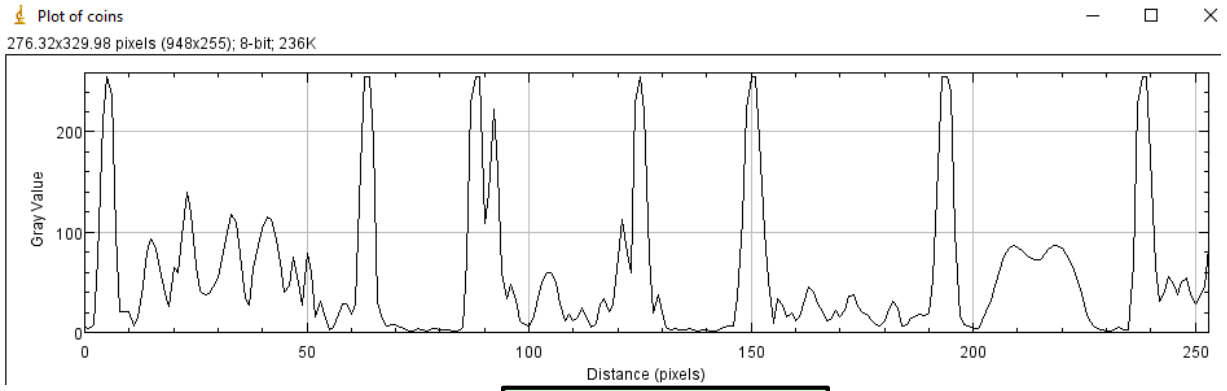
original image and 1D intensity profile



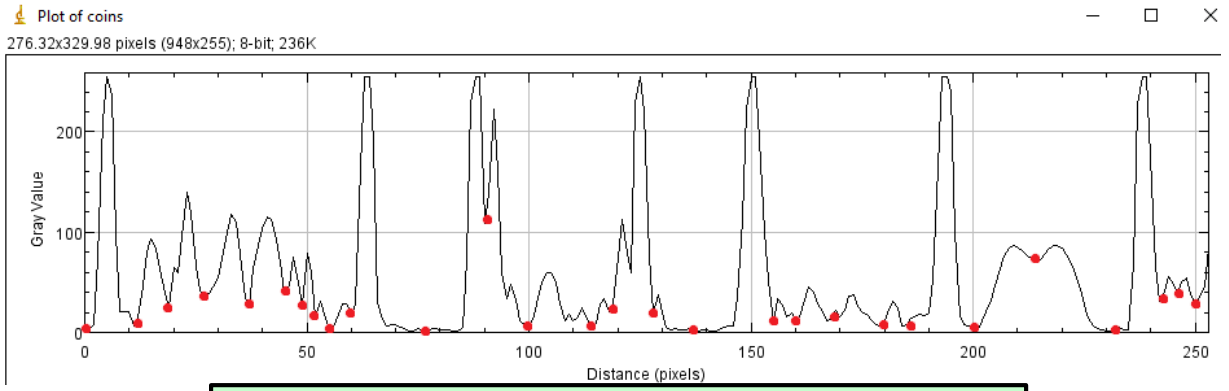
edges derived from input image and 1D intensity profile

Segmentation Methods cont'd

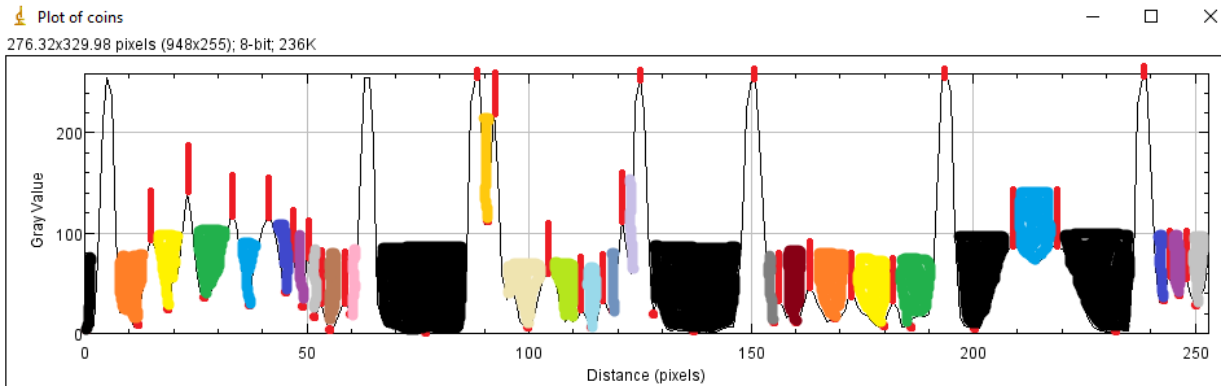
- **watershed transformation cont'd**:



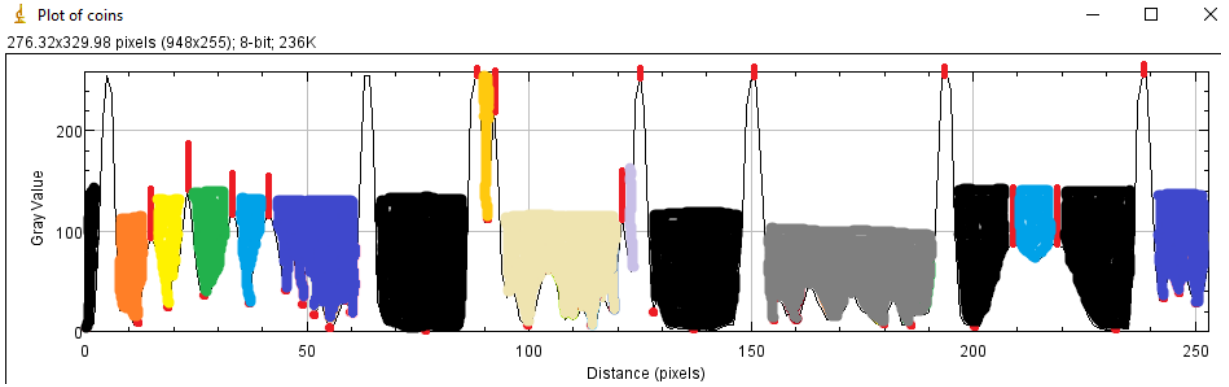
(1) initial profile



(2) local minima detection as seed candidates



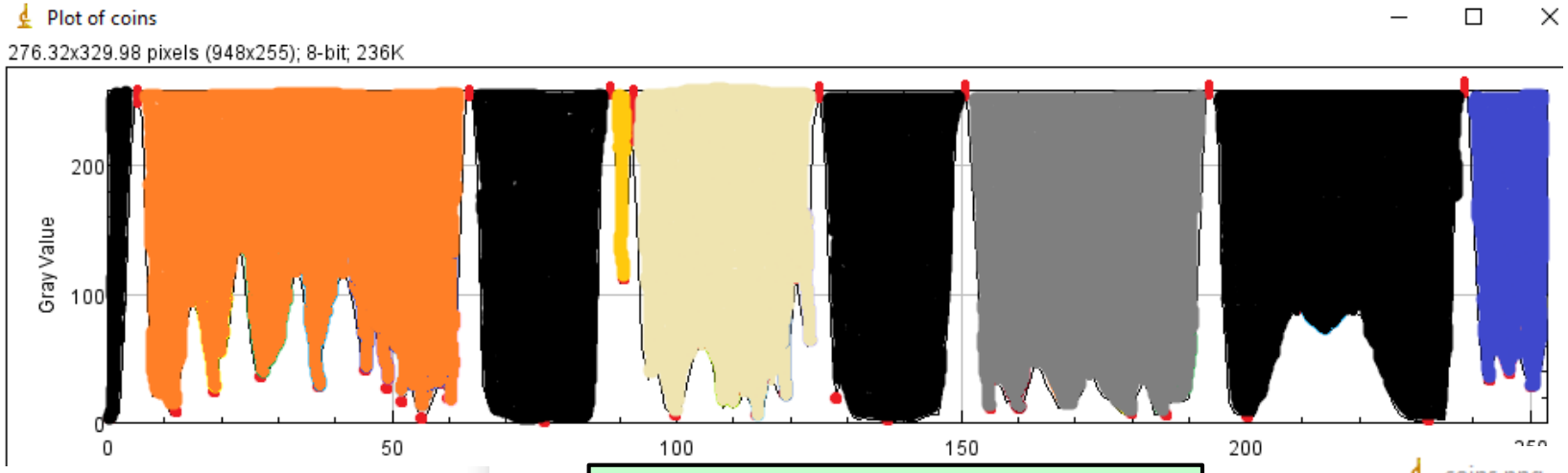
(3) starting at the seed minima, the regions are flooded, i.e. incremented. In case of collision with neighboring region, a watershed barrier of pre-specified height is placed as separation wall



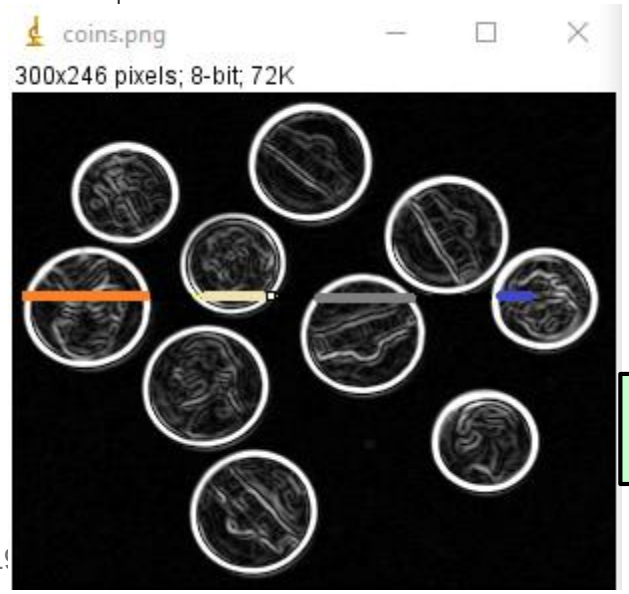
(3) in case of increased water level, some of the water sheds are surmounted, thus leading to region merge

Segmentation Methods cont'd

- **watershed transformation cont'd**:

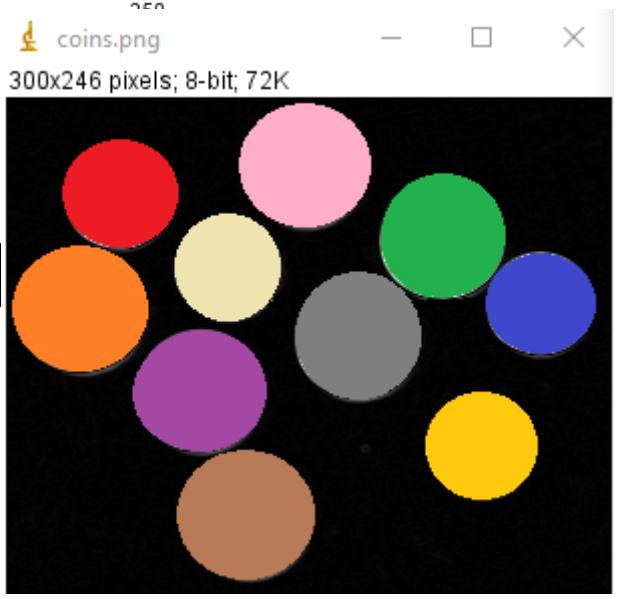


(5) final result only contains the clearly separated regions



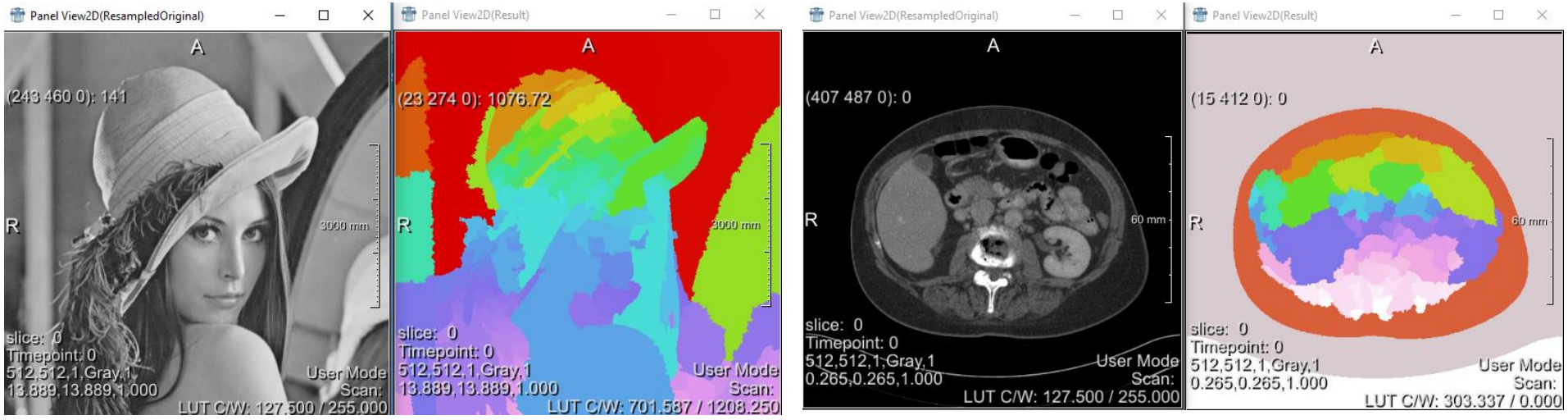
final 1D profil displayed in the original input image

result of 2D watershed applied



Segmentation Methods cont'd

- **watershed transformation discussion**:
  - quality of results strongly depend on input image signal-to-noise ratio. In case of artifacts or weak borders, the "*weakest*" link, i.e. local pixel area, determines, if two neighboring basins can be separated or if they are misleadingly merged
  - thus, incorporation of intensity profile similar to confidence connected thresholding can significantly stabilize the results [Zwettler 2014].
  - besides, length of the shared border of two neighboring regions need to be considered too. The smaller the conjoined path, the stricter the metrics for potential region merge [Zwettler 2014].
  - processing 3D data, the signal-to-noise ratio is even more crucial regarding results as for hull-shaped growing regions a local breach, i.e. image artifacts, is very likely.



Segmentation Methods cont'd

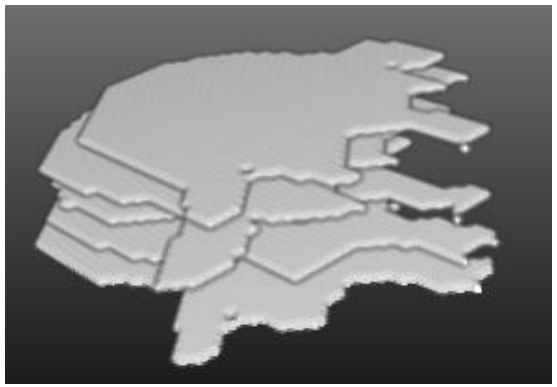
- shape- and model-based segmentation approaches
  - **Live-Wire Contour Detection**: semi-automated specification of regions by placing control points along the contour of the target region. Path between the control points minimized according to cost function, balancing between minimal path length and minimal gradient difference along the path. Cost function trained from reference segmentations [Schenk and Prause 2001].
  - **Deformable Models / Active Contours (snakes):** cost function balances between internal and external energy, e.g. being attracted by target object borders while preserving a certain shape [McInerney and Terzopoulos 1996].
  - **Statistical Shape Models**: trained from a huge set of adequate reference segmentations by calculating the average shape and the "*modes of variation*" for dealing with characteristic deviation of the target object's morphology [Cootes et al. 1992].
  - **Active Appearance Models:** advancement of deformable models by incorporating intensity profile match of the regions formed by the active contour control points [Cootes et al. 1998].
  - **Level-Sets**: advancement of deformable models, as changes in topology and the number of control points required for shape description at each iteration are implicitly handled [Osher and Sethian, 1988].

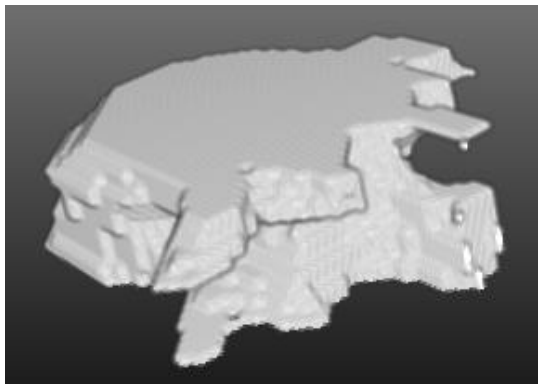
Segmentation Methods cont'd

- **Live-Wire** segmentation
  - increasing the necessary level of user interaction, even more difficult structures and shapes can be segmented in an semi-automated way.
  - control points delineating the target object's shape are provided by the user
  - the Live-Wire algorithm then progressively fills up the polyline-path between the gradients (*Laplace, Sobel, LoG,... all with adjustable weight*), thereby minimizing an energy function
  - The energy function thereby balanced the criterions "short path length" and "minimal energy", i.e. vertical movement in case of gradient changes.
    - thus, the contour will rather stay on the edges as going down the slope and up again would be too costly.
    - *imagine a wanderer reaching several summits on a tour. He will rather stay up on the back of the mountain and accepting a slight detour instead of directly going down the valley each time*

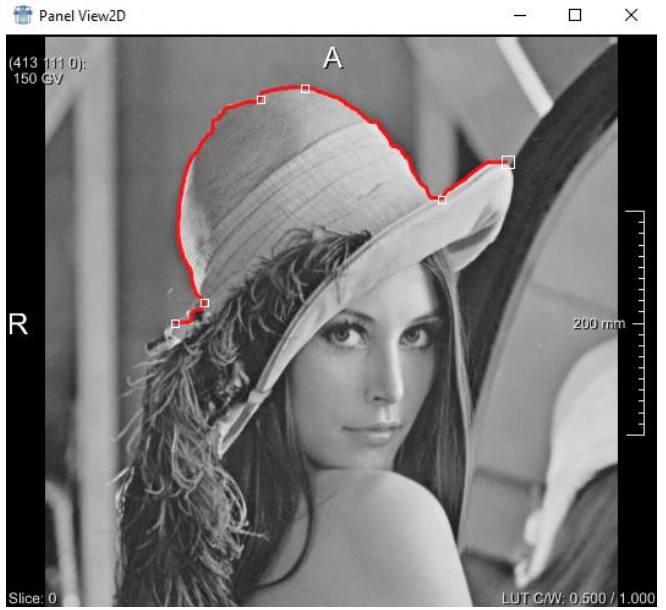
Segmentation Methods cont'd

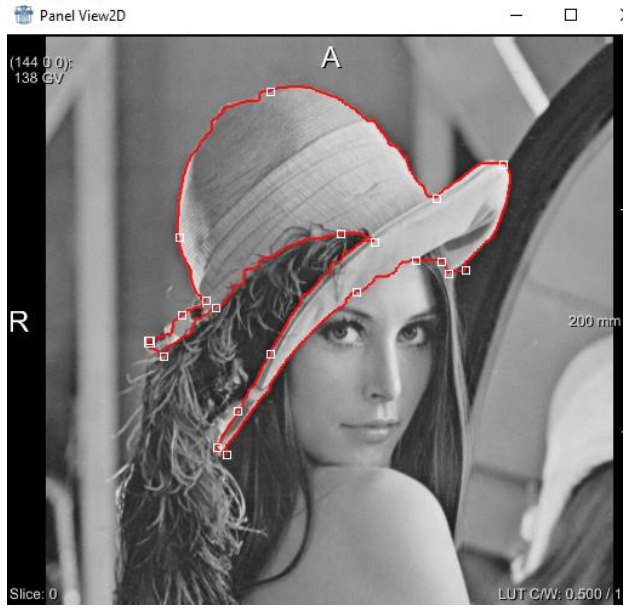
- between the control points the path is automatically optimized and the cost function trained/updated ➔ for next frames less control points required





explicitly defined and interpolated slices



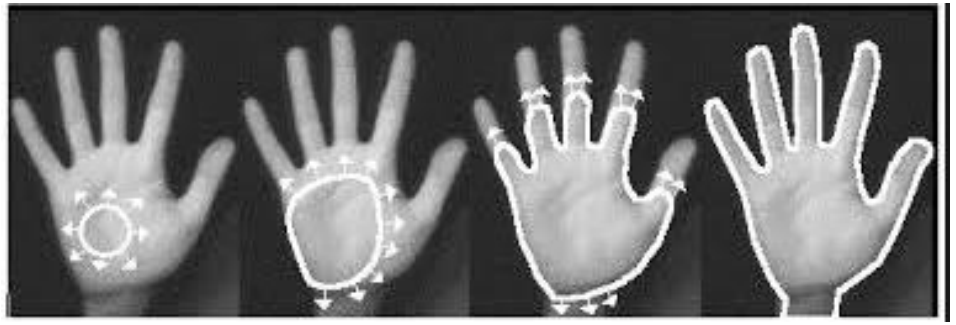


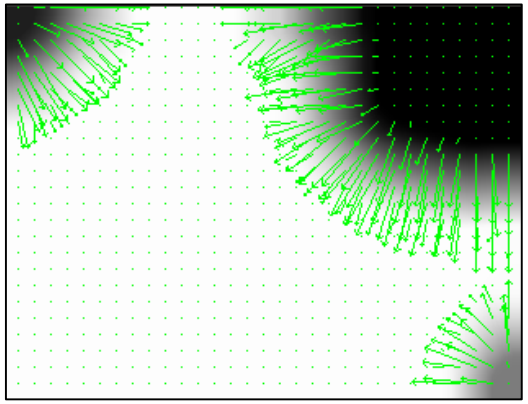
few control points sufficient for contour-segmentation

in case of processing large tomographic 3D datasets, interpolation is applicable on left-out slices utilizing subsequent path optimization

Deformable Models, Active Snakes

- active snake defined by a set of control points $v = \langle v_1, v_2, \dots, v_n \rangle$ with n control point coordinates $v_i = \langle x_i, y_i \rangle$ represent an energy minimization spline.
- $E_{snake} = E_{internal}(v) + E_{external}(v) + E_{constraint}(v)$
  - external energy as attraction of the object contour, e.g. via gradients weighted by γ as $E_{external}(v) = -\gamma \left| grad\big(I(v)\big) \right|$ for image I.
    - defined by edge gradients, intensity match or derived force fields
  - internal energy
    - defined by elasticity weight α (path length) and stiffness weight β (direction changes in the path) in course of stretching and bending.
    - alternatively: similarity with pre-defined shape
  - constraint energy
    - higher-level understanding of image and/or the shape
  - gradient descent search direction for optimization of the contour, i.e. minimization of the energy, at each point and iteration
  - problems to solve: *how to initialize the contour? how to handle changes in topology and changes in growth w.r.t. the discrete control points*





vector field as advection towards object contours

Statistical Shape Models

- generation of shape-describing a priori models covering the target object's characteristic **shape** and characteristics shape **variation**.
- therefore, a set of (manually) pre-segmented training datasets is required, covering the desired shape variability.
- process to train a statistical shape model (SSM):
  - (1) prepare a set of representative and accurate reference-segmentations
  - (2) extraction of robust shape-describing and correlating landmarks.
    - the number of landmarks must be the same for each dataset
    - thus, automatic landmark extraction is often utilized, generating pseudo-landmarks by performing equiangular raytracing from region centroid or region skeleton
    - in case of n landmarks p_i all as 3D coordinates $p_i = \langle x_i, y_i, z_i \rangle$ the problem dimensionality, i.e. size of each particular shape vector $\vec{F} = (x_1, y_1, z_1, x_2, y_2, z_2, \dots, x_n, y_n, z_n)$ has to be specified with $n \cdot 3$
  - (3) application of Procrustes alignment to allow modelling of a configurable a priori shape invariant of affine transformations such as *position*, *scale* and *orientation*
    - all landmark coordinates are thereby scaled to a normalized coordinate system with size $[-1; 1]$ per dimension and centroids located at the origin (ad hoc translation).
    - distances of the landmarks from centroid are normalized to $\sigma = 1$, thus equalizing differences in scale
    - registration of the shapes as optimization of the orientation thereby evaluating ICP distance metrics.

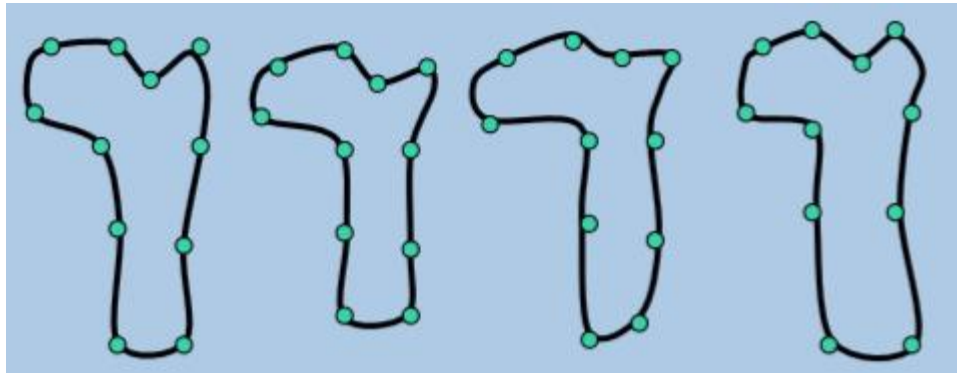
Statistical Shape Models cont'd

- (4) calculation of the mean shape vector $\bar{F}$ from *s* reference datasets with $\bar{F} = \frac{1}{s}\sum_{j=1}^{s} F_j$
- (5) application of **principal components analysis** to derive most relevant m component vectors u_i with $m \ll n$.
  - therefore, first the mean vector $\bar{F}$ is subtracted from all form vectors F.
  - then, the covariance matrix is calculated for all form vector pairs $(F_i, F_j)\ \forall i \forall j, i \leq n \wedge j \leq n \wedge i \neq j$
  - calculate (unit) eigenvectors and eigenvalues from covariance matrix Σ.
  - the first principal component is the eigenvector with highest associated eigenvalue, i.e. orthogonal axis corresponding to the main direction of variation
  - sorting the eigenvectors by eigenvalues, one gets the first *n* principle components.
  - *Singular Value Decomposition* (**SVD**) applicable for direct calculation of eigenvalues and eigenvectors
- (6) the statistical shape model is finally defined as $F = \bar{F} + \sum_{i=1}^{m} c_i \cdot u_i$ with $\mathrm{u_i}$ as eigenvector (mode of variation) and coefficient distributed $-3\sqrt{\lambda_\mathrm{i}} \leq \mathrm{c_i} \leq 3\sqrt{\lambda_\mathrm{i}}$ according to particular eigenvalue λ_i.
  - each component vector u_i necessitates individual weighting coefficient c_i to allow precise reconstruction of arbitrary input object shapes.
- (7) application of a statistical shape model for individual object shape approximation:
  - normalize scale, locate at centroid.
  - then optimize the registration parameters $rot, u_1 \ldots u_m$ for best shape match

Statistical Shape Models cont'd

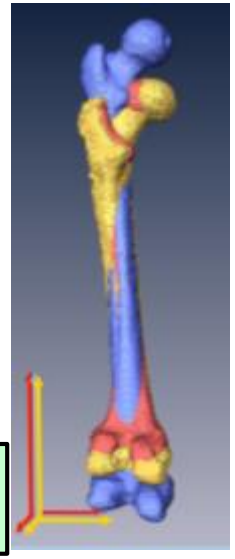
- Statistical Shape Modeling in Medical Image Analysis [Reyes 2010] : orthopedic implant design for left femur

human bones with anatomical variability

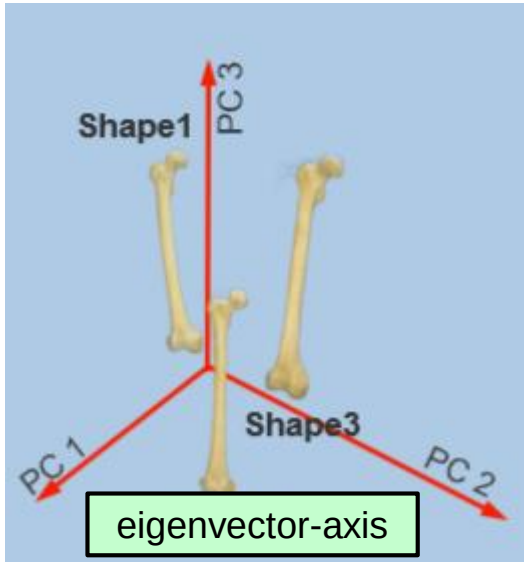


extraction of corresponding anatomical landmarks

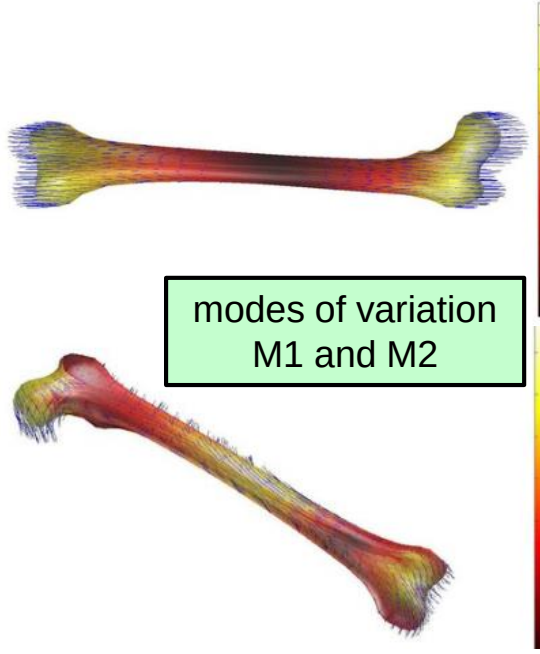
calculation of mean shape vector



Procrustes alignment



eigenvector-axis



modes of variation M1 and M2

Mathematical Morphology

- after performing binary segmentation on image ***A***, quality of results can be further improved utilizing morphological operators for post-processing
  - input data thereby expected to be [0;1] or [0;255] with 1 and 255 respectively as “foreground”, i.e. segmented object, and 0 as “background”.
  - for application of morphological operators, a filter mask, denoted as structuring element ***S***, is required.
- the two basic operators are erosion and dilation:
  - **erosion**: foreground is equally shrinked by small radius, similar to removing one single skin layer of an onion **B=A ⊖ S**.
  - **dilation**: operator contrary to erosion: the foreground grows by a small radius **B=A ⊕ S**.

Mathematical Morphology cont'd

- morphological operators are often sequentially executed, thus some considerations are required:
  - dilation:
    - **A⊕S == S⊕A** kommutative
    - **(A⊕B)⊕C == A⊕(B⊕C)** associative, chain rule
  - erosion:
    - **A⊖S != S⊖A** **not** kommutative
    - **(A ⊖ B) ⊖ C == A ⊖(B⊕C)** chain rule
  - generally, one erosion run utilizing a specific structuring element ***S*** will marginally remove more mass compared to the mass that is added via dilation with the same structuring element.
  - structuring element design affects results.
    - generally, applying small structuring elements more often will lead to better performance compared to applying a larger structuring element once.
    - larger structuring elements allow better approximation of e.g. circular growth

0	1	0
1	1	1
0	1	0

N_4

small disk 7x7

0	0	1	1	1	0	0
0	1	1	1	1	1	0
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
0	1	1	1	1	1	0
0	0	1	1	1	0	0

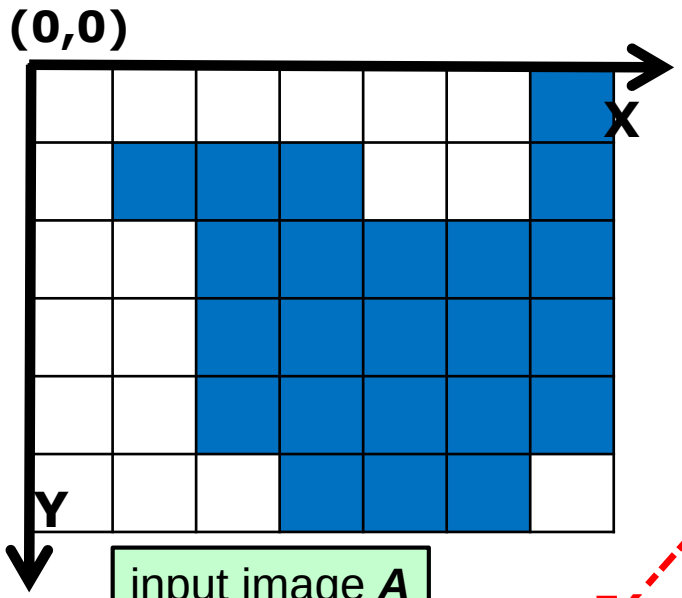
Mathematical Morphology cont'd

- **erosion**
  - at each foreground (FG) hot-spot position in image ***A*** with coordinates in *(x,y)*, the structuring element ***S*** with radius *r* and coordinates *(k,l)* is placed. The hot spot is conserved only, if all set neighbour positions in ***S*** match the relative foreground positions in ***A***.

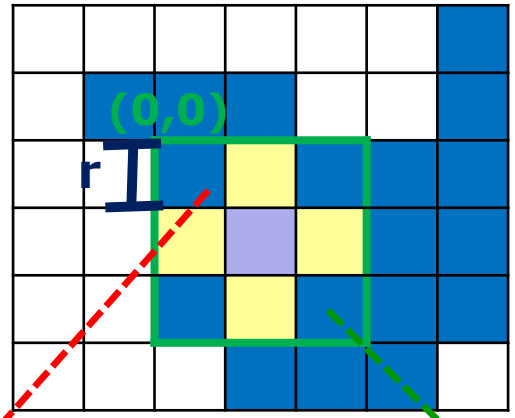
$$I'(x,y) := \begin{cases} 1 & \forall_{k=x-r}^{x+r} \; \forall_{l=y-r}^{y+r} \; I(k,l) \geq S(k-(x-r), l-(y-r)) \\ 0 & else \end{cases}$$

0	1	0
1	1	1
0	1	0

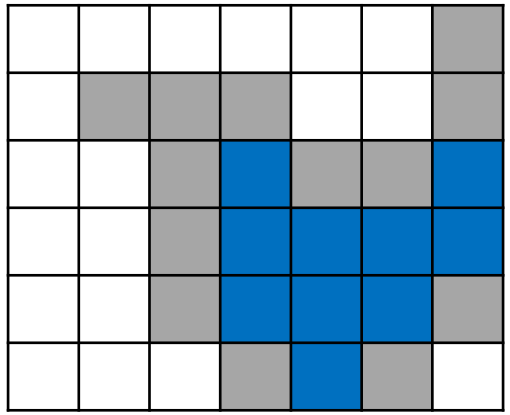
structuring element N_4



input image ***A***



S evaluated at all FG position in ***A***



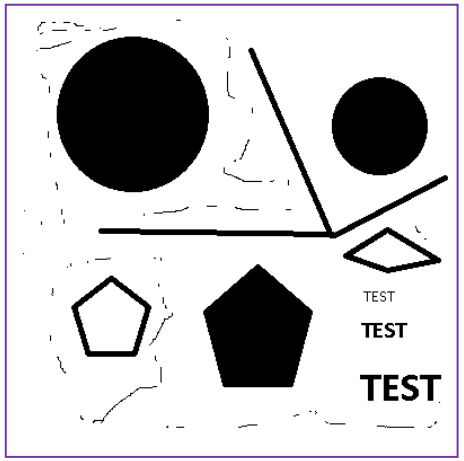
result image ***B***

I(x,y)

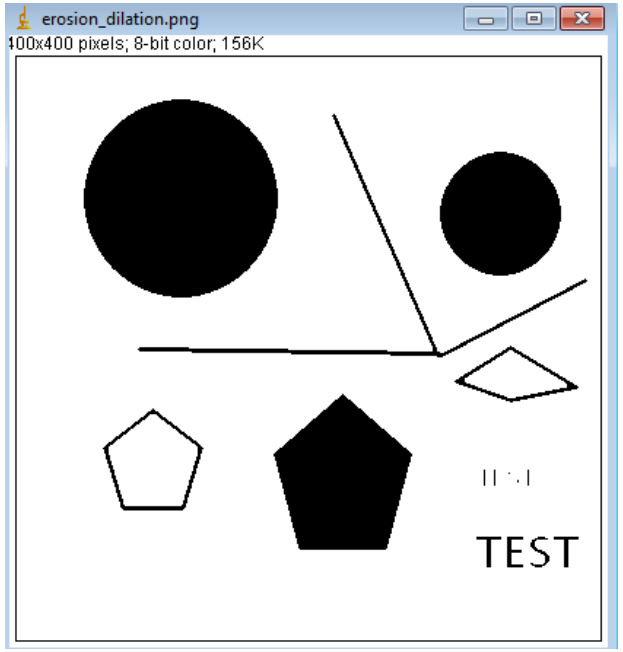
S(k,l)

Mathematical Morphology cont'd

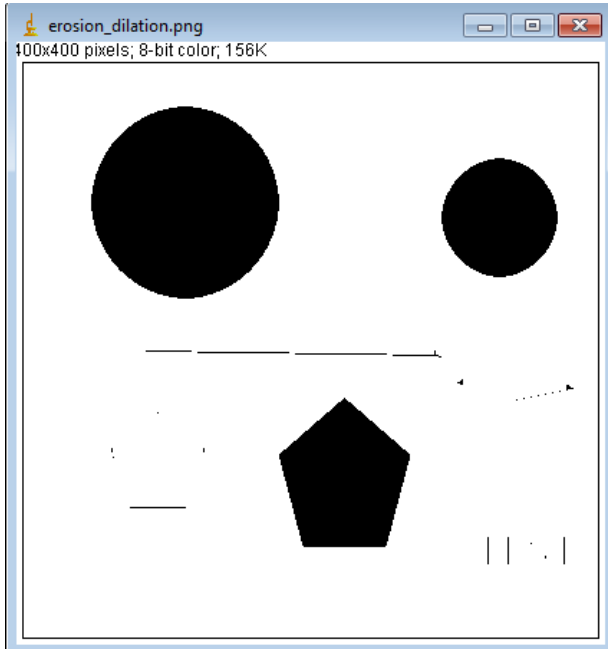
- **erosion** cont'd
  - artefact removal
  - removal of small connections



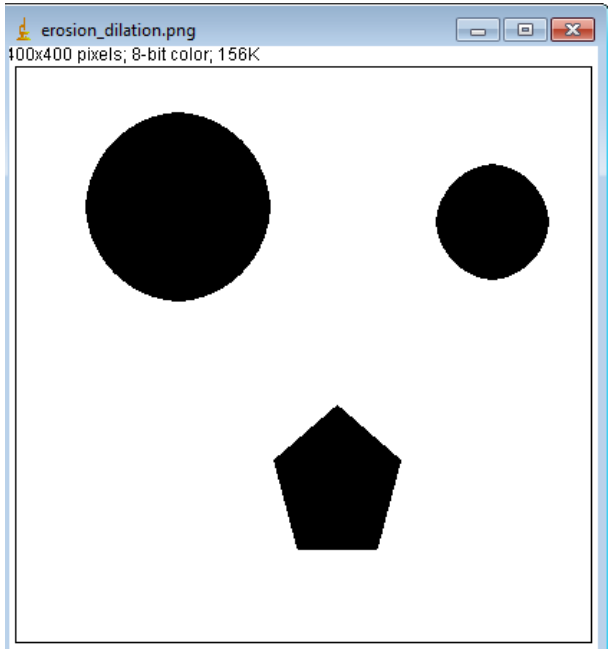
input image ***A***



result after iteration #1 (colours reversed)



result after iteration #2



result after iteration #3

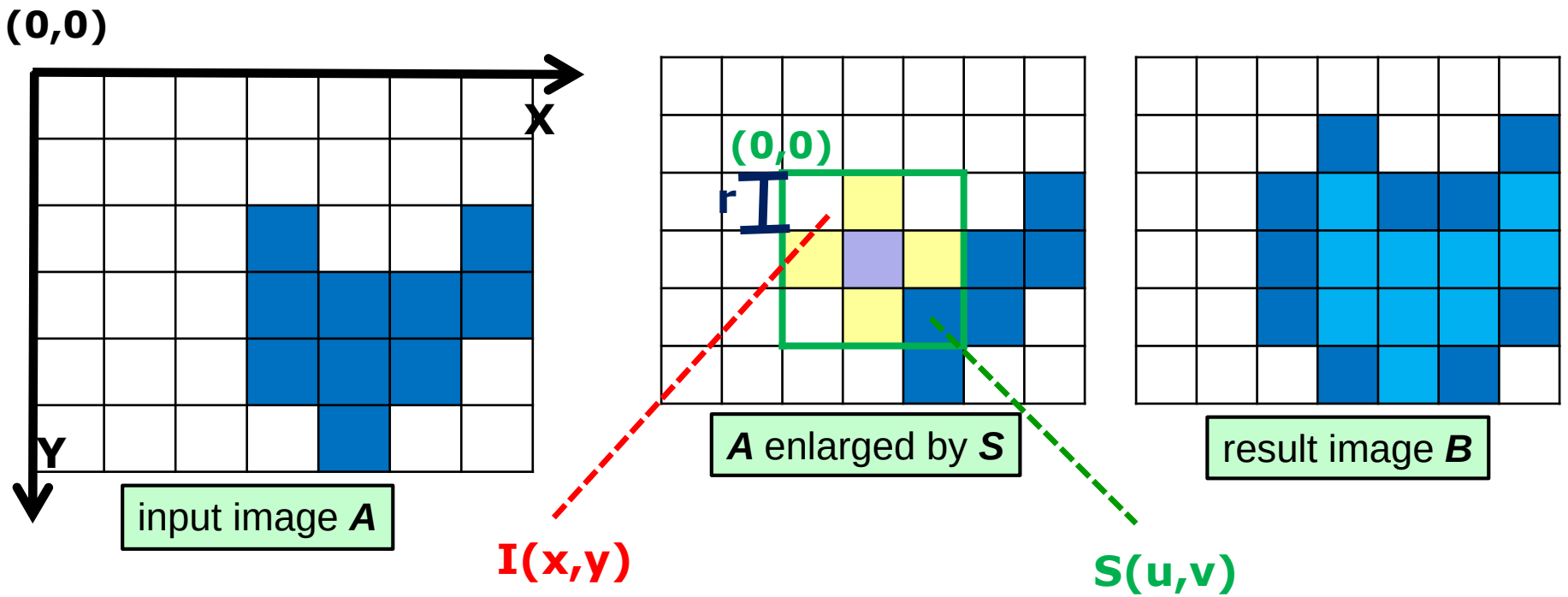
Mathematical Morphology cont'd

- **dilation**
  - at each foreground (FG) hot-spot position in image ***A*** with coordinates in *(x,y)*, the structuring element ***S*** with radius *r* and coordinates *(k,l)* is placed. All set neighbour positions in ***S*** are assigned a foreground value.

$$I'(x,y) := \begin{cases} 1 & \exists_{k=x-r}^{x+r} \exists_{l=y-r}^{y+r} I(k,l) = 1, S(k-(x-r), l-(y-r)) = 1 \\ 0 & else \end{cases}$$

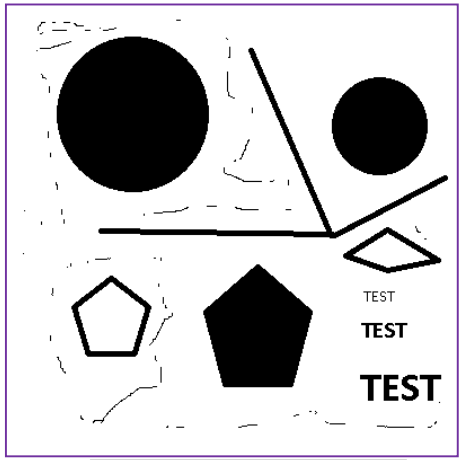
0	1	0
1	1	1
0	1	0

structuring element N_4

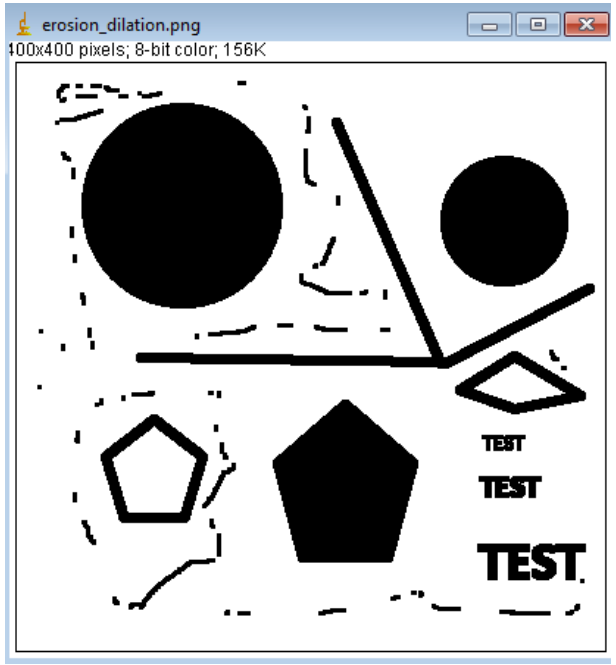


Mathematical Morphology cont'd

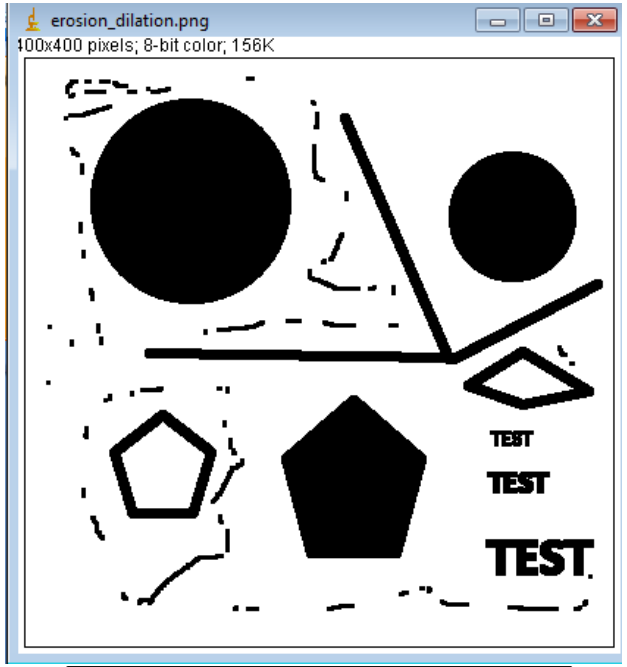
- **dilation** cont'd
  - fills up small holes
  - interconnects neighbouring structures



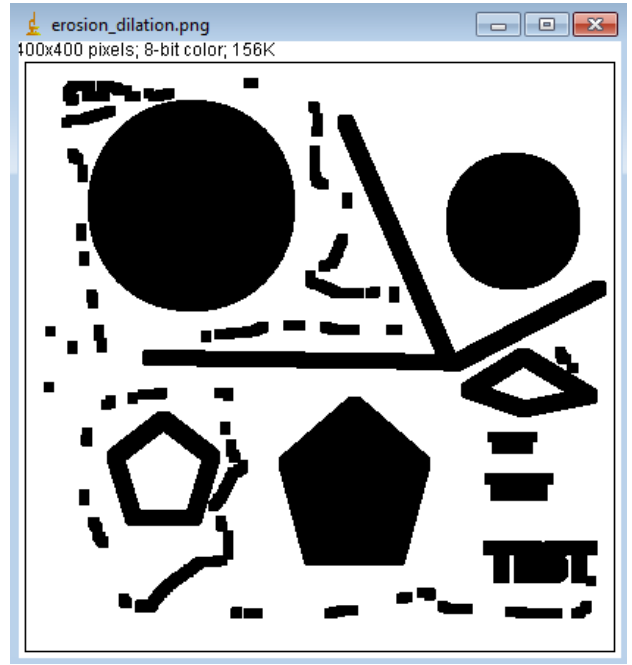
input image ***A***



result after iteration #1 (colours reversed)



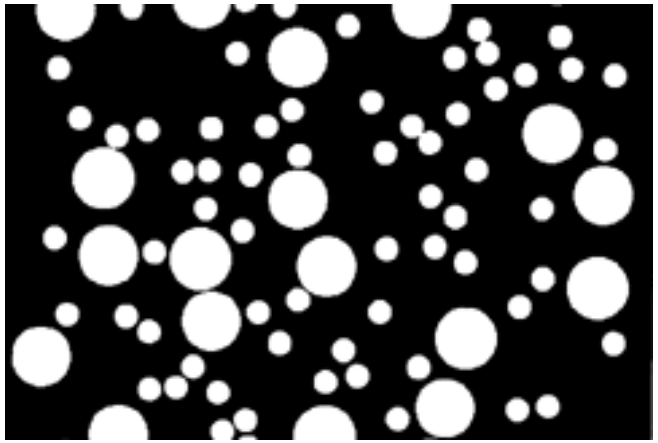
result after iteration #2



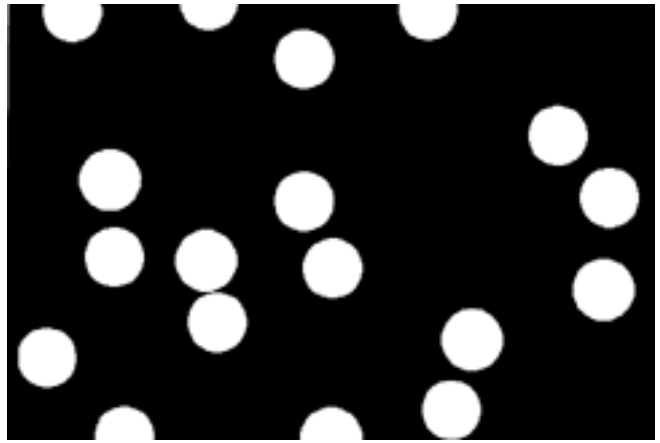
result after iteration #3

Mathematical Morphology cont'd

- **opening**
  - operation composed of several erosion and dilation runs (*generally approx. equal number*) utilizing structuring element ***S***
  - first erosion, then dilation
  - **$B \circ S = (A \ominus S) \oplus S$**
  - small artefacts are removed
  - small connections between structures are bursted



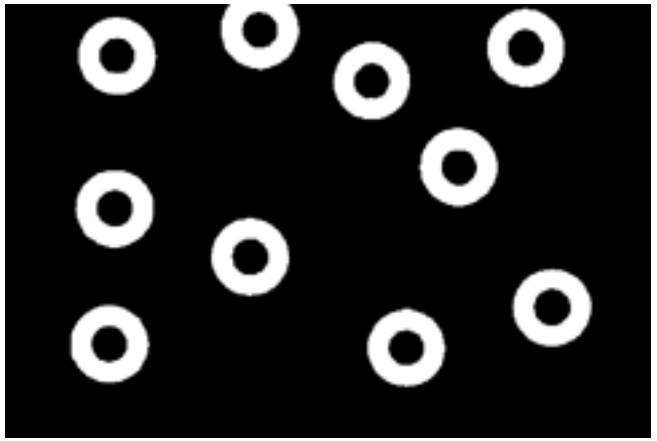
input image ***A***



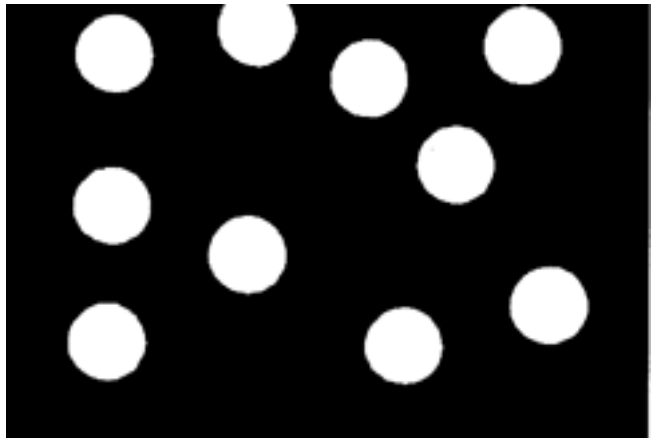
result image ***B***

Mathematical Morphology cont'd

- **closing**
  - operation composed of several erosion and dilation runs (*generally approx. equal number*) utilizing structuring element ***S***
  - first dilation, then erosion
  - **$B \bullet S = (A \oplus S) \ominus S$**
  - small holes get filled



input image ***A***



result image ***B***

Mathematical Morphology cont'd

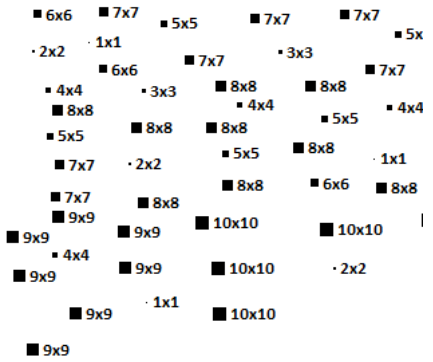
- **hit-or-miss operator**
  - while for erosion and dilation structuring element S the 0-values were irrelevant, for hit-or-miss operation they are of high relevance to allow pattern matching
  - the hit-or-miss mask contains values of **0**,**1** and **X**.
  - if evaluating the operator at a specific hot-spot position of input image ***A***, a match is given if:
    - in case of **1**, the image shows a *foreground* value at that position
    - in case of **0**, the image shows a *background* value at that position
    - in case of **X** – neutral as "don't care"
  - exact match with large operators not robust w.r.t. noise level, thus some match-confidence might be introduced

0	1	0
1	1	1
0	1	0

cross detector

X	1	X
0	1	1
0	0	X

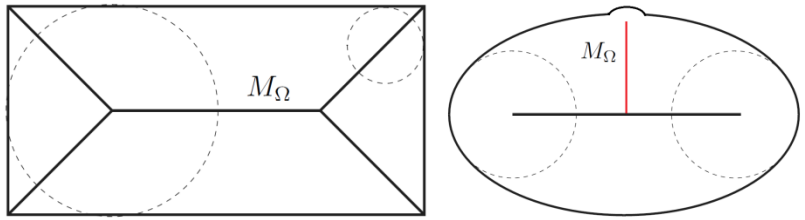
detector for lower left corner



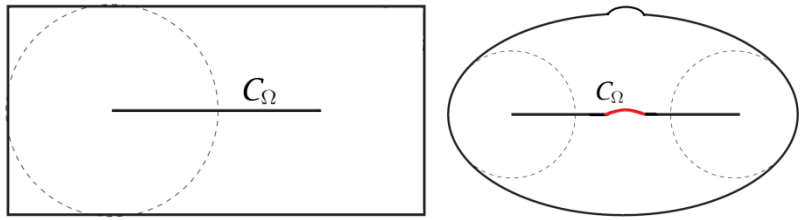
possible task: detect all black 3x3 squares

Mathematical Morphology cont'd

- **extraction of inner shape of 2D segmentations**
  - after binary segmentation, the inner shape can be derived via *skeletonization* or *medial axis*.
  - **skeletonization:** circles of arbitrary radius are virtually enclosed inside the shape. If they are inside the shape and touch at least 2 border pixels of the shape, the central point of the circle is part of the skeleton. Can be calculated based on *Euclidean Distance Maps*.
  - **medial axis**: result of iterative thinning process. Erosion is applied until only the inner part of thickness 1pixel remains. No erosion in case of potential topographic break-ups or foreshortening.



skeletonization

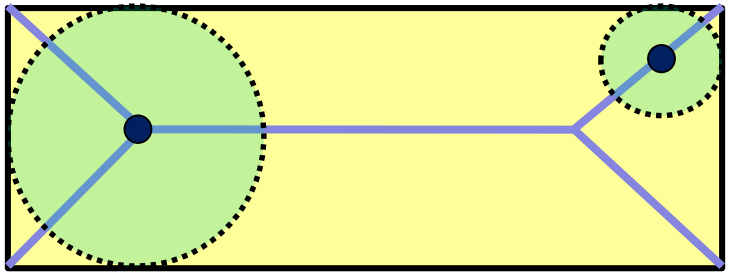


medial axis extraction

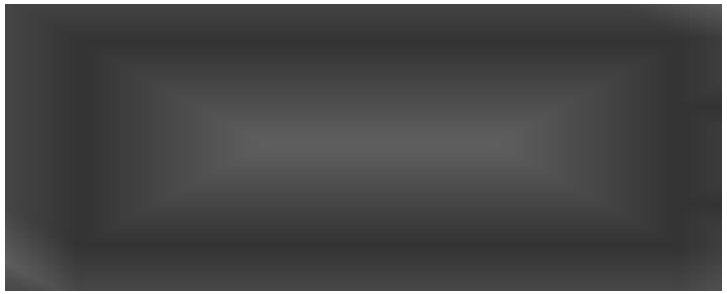
Mathematical Morphology cont'd

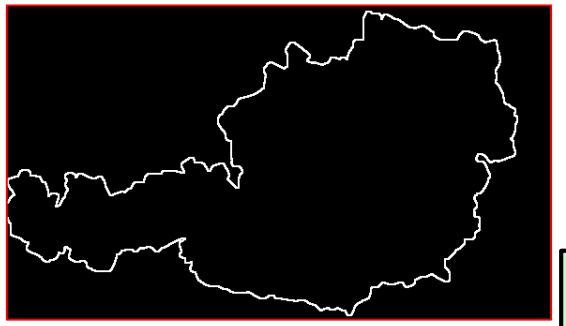
- **skeletonization**
  - a skeleton of a binary shape object ***O*** covers all pixel positions, from where the circumference at any radius is within the shape exactly touching two or more border pixels.
    - can be calculated utilizing Chamfer Distance Maps, e.g. Euclidean distance metrics
    - if specific radius is stored for any of the skeleton elements, the original shape can be reconstructed. Thus, applicable as compression strategy too.
    - *disadvantage*: no constant thickness of 1 is guaranteed and topography cannot be conserved, c.f. rectangle or *L/H-shape*

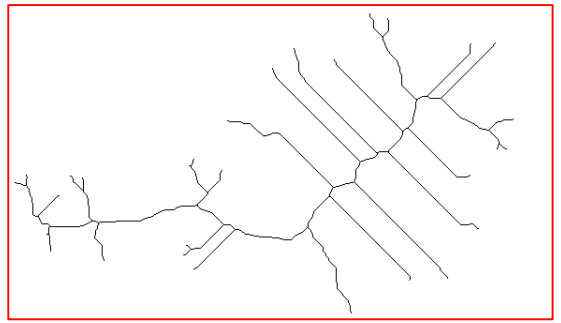
skeleton calculation

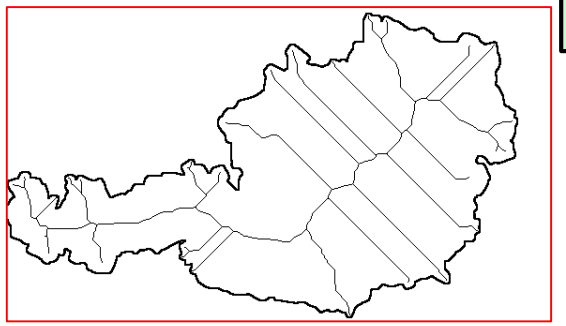


distance map





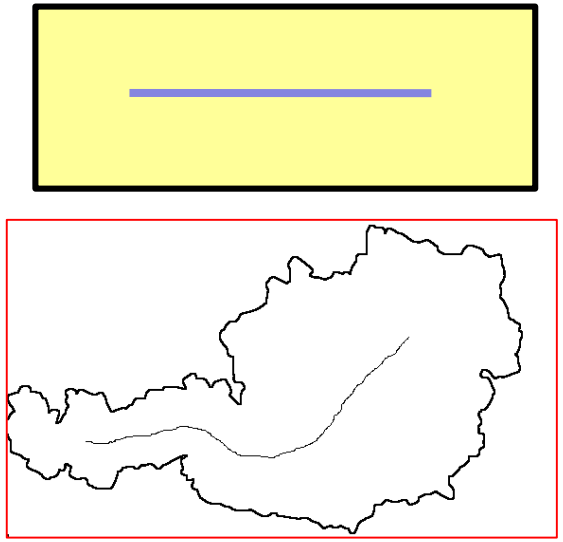




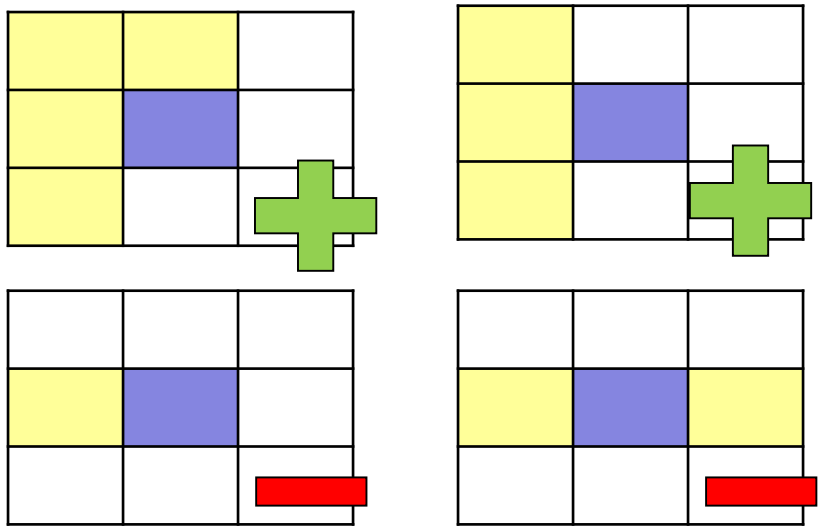
skeleton of Austria

Mathematical Morphology cont'd

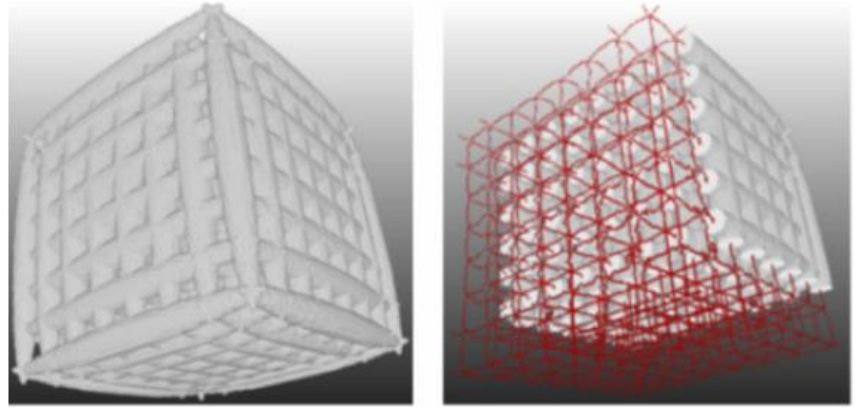
- **thinning / medial axis**
  - e.g. solvable utilizing Randomized Erosion [Zwettler et al. 2010]
    - thereby, the surface gets eroded until the medial axis only remains
    - constant thickness of 1pixel and conserved topography is guaranteed
    - original shape cannot be reconstructed
    - no erosion in case of foreshortening (end), splitting the graph or grabbing into the surface



medial axis of Austria



neighbourhood constellations where erosion is allowed/prohibited



randomized erosion perfectly applicable on 2D and 3D data

Bibliography

- G. Zwettler, W. Backfrieder, R. Swoboda, F. Pfeifer – “Fast Medial Axis Extraction on Tubular Large 3D Data by Randomized Erosion”. In: *Lecture Notes in Communications in Computer and Information Science (CCIS)* - Springer Verlag, 2010, pp. 97-108
- G. Zwettler – “*General Model-Based Segmentation Strategy for Holistic Analysis of Tomographic Medical Image Data in 3D Radiology”.* In: PhD Thesis, University of Vienna, Austria, 2014, pp.1-273.
- S. Osher, J.A Sethian – “Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations”. In *Journal of Computational Physics* 79, 1988, pp.12-49.
- T.F. Cootes, C.J. Taylor, D.H. Cooper and J. Graham- “*Training Models of Shape from Sets of Examples”*. In Proceedings of the British Machine Vision Conference, Leeds, U.K., 1992, pp. 9-18.
- T.F. Cootes, G.J. Edwards and C.J. Taylor – “*Active Appearance Models”*. In Proceedings of the 5th European Conference on Computer Vision, 1998, pp. 484-498
- T. McInerney and D. Terzopoulos – “Deformable Models in Medical Image Analysis : A Survey”. In *Medical Image Analysis* 1 (2), 1996, pp.91-108.
- A. Schenk and G.P.M Prause – “Optimierte semi-automatische Segmentierung von 3D Objekten mit Live-Wire und Shape-Based-Interpolation”. In *Bildverarbeitung für die Medizin,* 2001, 202-206.

UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

Image Processing

Compression

DI(FH) Dr. Gerald Adam Zwettler, MSc, gerald.zwettler@fh-hagenberg.at
Lector of Computer Graphics, Image Processing and Computer Vision

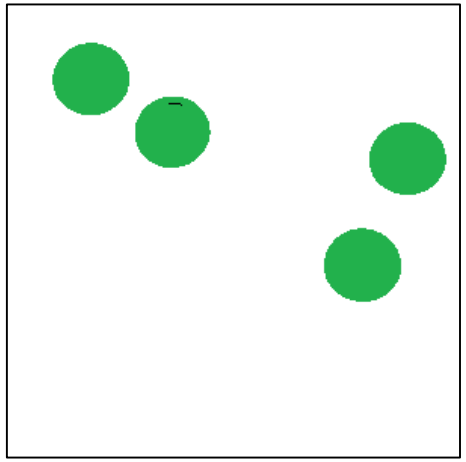
Representation of Segmented Images

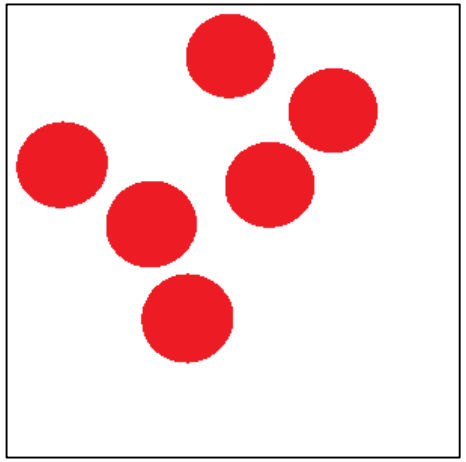
- segmentations provide additional information on the original input image data. Thus, strategies are required to prevent from multiplying the required data amount
  - (a) store each segmented class/object in a separate image
  - (b) store all segmented classes/objects in a single image
  - (c) store pixel-positions per segmented class/object instead
  - (d) store contour of segmented objects as compressed chain codes only

url link, bold, [8pt]

Representation of Segmented Images

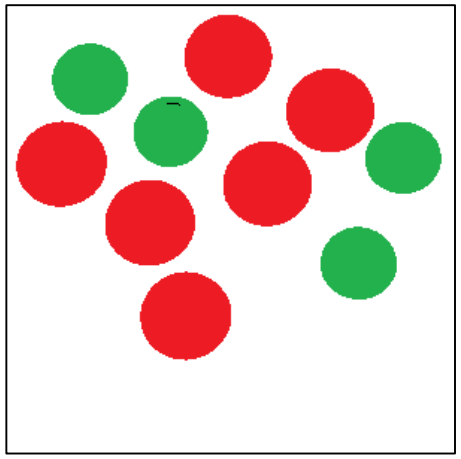
- (a) store each segmented class/object in a separate image
  - easy to process, as no encoding/decoding required
  - segmentation image data of same size as the original input image
  - scalar range might not be changeable for most image formats
  - required amount of data: `D = I*(n+1)` with `n`… number of separate segmentations
  - E.g. image ***I*** with 512x512 pixels and scalar type int16 (2 bytes) ➔ 512kB for original image and 512kB for each separate segmentation. In case of two segmentations, the amount of necessitated data is increased to 1.5MB





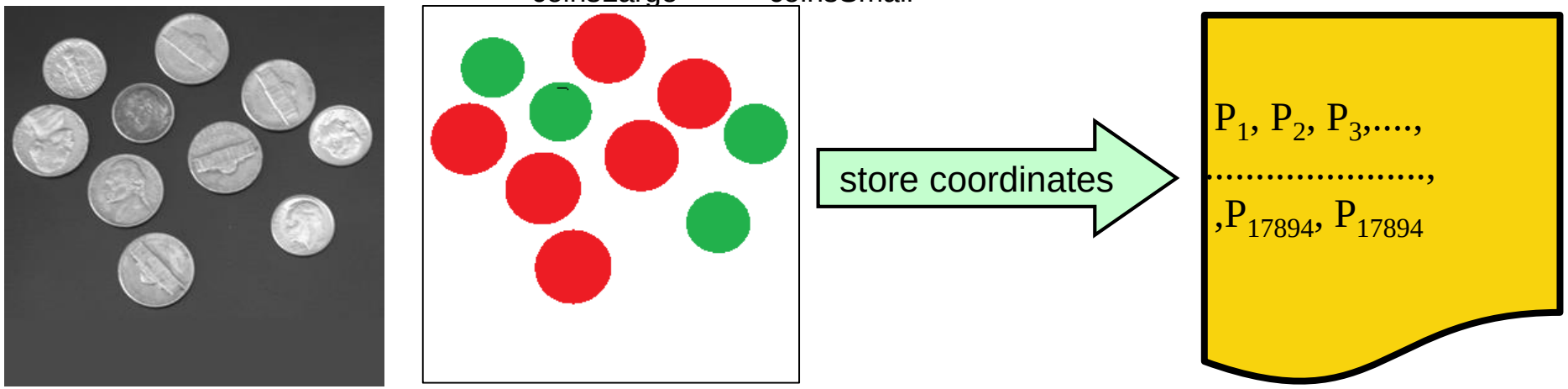
Representation of Segmented Images

- (b) store all segmented classes/objects in a single image
  - easy to process, as no encoding/decoding required
  - segmentation image data of same size as the original input image
  - scalar range might not be changeable for most image formats
  - required amount of data: `D = I*2` as unique indices for all segmentations within scalar range
  - the segmented regions must not be partially congruent to ensure valid indices and masking
  - E.g. image ***I*** with 512x512 pixels and scalar type int16 (2 bytes) ➔ 512kB for original image and 512kB for two segmentations, the amount of necessitated data is increased to 1MB



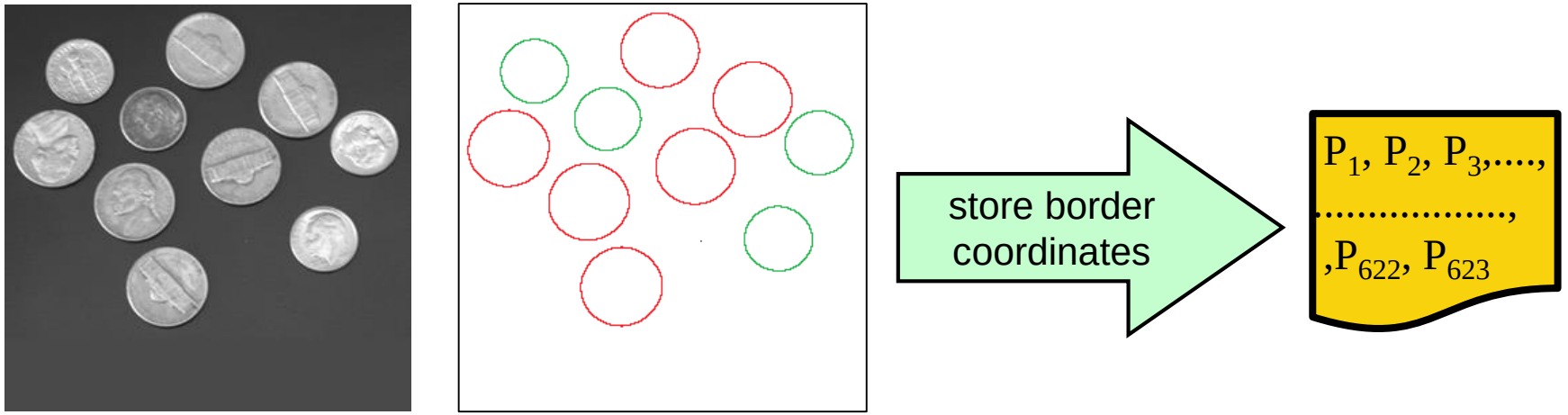
Representation of Segmented Images

- (c) store pixel-positions per segmented class/object instead
  - effective strategy if the segmented regions only cover a small ratio of the total pixel count
  - time-consuming encoding/decoding required before the images can be processed/visualized
  - overhead of storing two coordinates instead of 1 scalar value needs to be considered w.r.t the necessary scalar datatypes (*2bytes per coordinate if image dimension expected to exceed 256 pixels*)
  - E.g. image ***I*** with 512x512 pixels and scalar type int16 (2 bytes) ➔ 512kB for original image
    - $|S_{coinsLarge}|$ = 10,120 *2 = 19.77KB additionally
    - $|S_{coinsSmall}|$ = 7,775 *2 = 15.19KB additionally
    - total segmentation overhead: $|S_{coinsLarge}|+ |S_{coinsSmall}|$ = 34.96KB



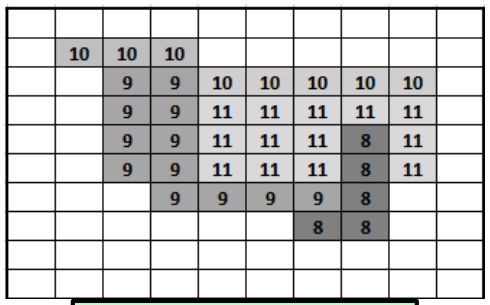
Representation of Segmented Images

- (d) store contour of segmented objects as compressed chain codes only
  - besides storing all voxels of a segmented region, it is adequate to persist the contour only
  - contour derived by *boundary tracing*. The original region can be reconstructed from the contour
  - e.g.
    - $C(|S_{coinsLarge}|)= 344 * 2$ ➔ 688 bytes additionally
    - $C(|S_{coinsSmall}|)= 279 * 2$ ➔ 558 bytes additionally
    - total: $C(|S_{coinsLarge}|)+C(|S_{coinsSmall}|)$=1.22KB additionally

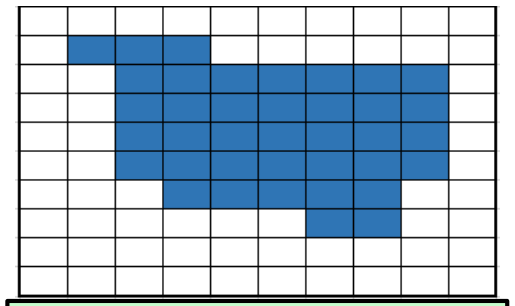


Representation of Segmented Images

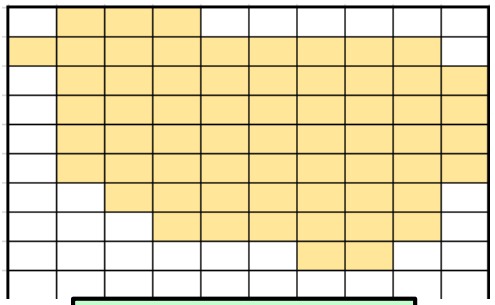
- processing 2D images, the contour contains the required information on the segmented regions
  - the contour of a region ***R*** is thereby defined as a unique path of pixels having both, foreground and background pixels in the neighbourhood.
  - binary contour representation required as input
  - the contour is extracted utilizing inner or outer boundary tracing ➔ list of sorted coordinates $C_i = (x_i, y_i)$ building up a distinct path.
  - inner boundary tracing: contour is part of segmented region [Sonka et al. 1993].
  - outer boundary tracing: all background pixels connected with one foreground element in N_4 defined as $N_4(C_i,C_j) := |C_i,C_j| = 1$. These pixels are tested performing inner boundary tracing algorithm and can be extracted utilizing mathematical morphology too: $A_{outerContour} = (A \oplus N_4) - A$



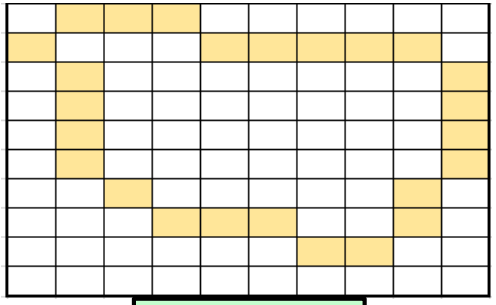
original image ***A***



A' binary in thresh [8;11]



A" dilated by N_4



$A_{outerContour}$

Representation of Segmented Images

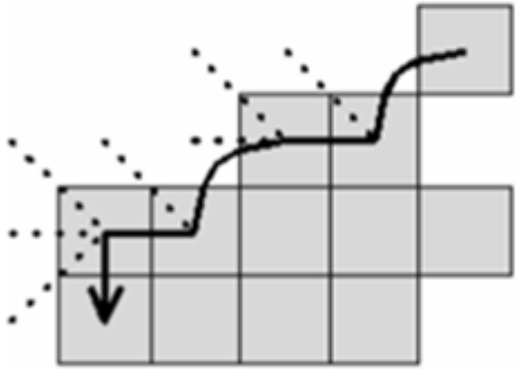
- Inner Boundary Tracing (N_8) algorithm:
  - (a) search from left-to-right, top-down until first foreground pixel is found
  - (b) search direction determined by variable ***dir***, initialized as ***dir*** = 7
  - (c) counter clockwise search until next foreground pixel is found
    - Search direction ***dir*** to utilize is updated from previous search result, namely
    - $dir' = (dir + 7)\ \%\ 8$, if ***dir*** is even
    - $dir' = (dir + 6)\ \%\ 8$, if ***dir*** is odd
  - (d) Add pixel C_i to result, update ***dir*** and continue with (c)
  - abort, if $C_i == C_1$ and $C_{i-1} == C_0$, thus start is reached. Checking both elements as narrow points or bifurcations might exist

Representation of Segmented Images

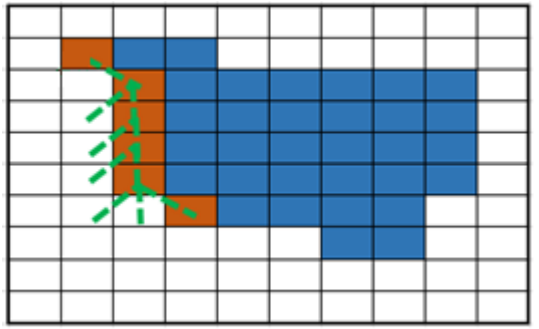
- Inner Boundary Tracing cont'd:
  - outer boundary traced by counter-clockwise search
  - all pixels tested during algorithm execution (dashed line) are element of outer boundary
  - Algorithm also applicable to N_4 adjacency, utilizing init ***dir*** = ***0*** and updating ***dir*** as $dir' = (dir + 3)\ \%\ 4$

search direction for N_4 adjacency

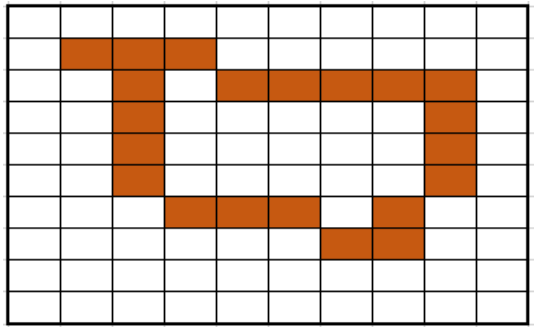
search direction for N_8 adjacency



boundary tracing for N_8 with dashed lines as ineffectively tested pixels



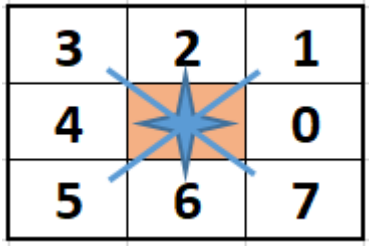
performing inner boundary tracing with N_8

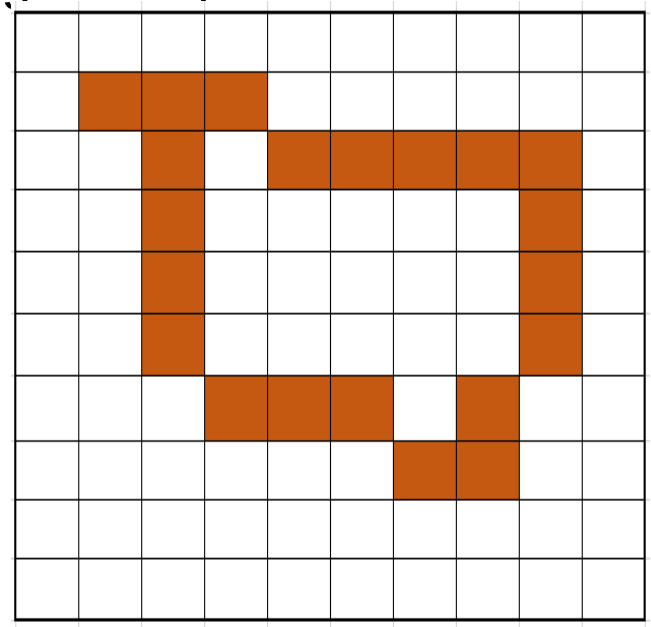


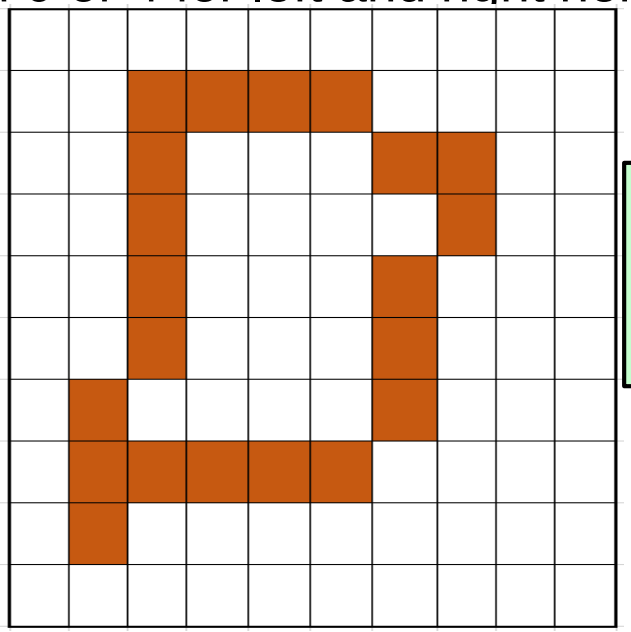
inner boundary result

Representation of Segmented Images

- Chain Codes
  - chain codes represent a numeric representation of 2D contours
  - each pair of Coordinates C_i, C_j element of the path with $N_8(C_i, C_j)$ is replaced by value according to adjacency chain code,
    e.g. 2 horizontally neighboring coordinates are valued 0 or 4 for left and right neighbor respectively







contour rotated by 90° leads to totally different chain code

matching parts:

chain code 7,6,6,6,7,0,0,7,0,2,1,2,2,2,4,4,4,4,3,4,4

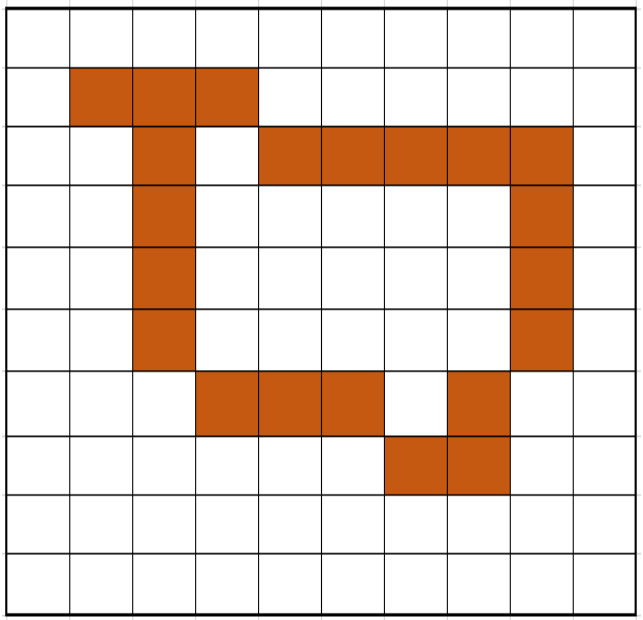
chain code 7,6,6,6,7,0,0,7,0,2,1,2,2,2,4,4,4,4,3,4,4

chain code 6,6,6,6,5,6,6,1,0,0,0,1,2,2,1,2,4,3,4,4,4

chain code 6,6,6,6,5,6,6,1,0,0,0,1,2,2,1,2,4,3,4,4,4

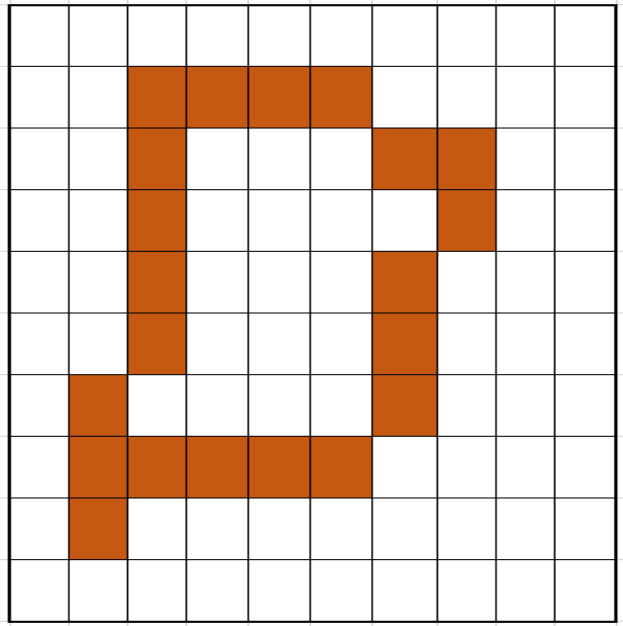
Representation of Segmented Images

- Predictive Chain Codes
  - chain codes calculated in common way but then predictive coding is applied calculating new sequence $d_i = (c_{i+1} - ci)\%8$
  - predictive sequence shorter by 1 element



chain code 7,6,6,6,7,0,0,7,0,2,1,2,2,2,4,4,4,4,3,4,4

predictive CC: **7,0,0,1,1,0,7,1,2,7,1,0,0,2,0,0,0,7,1,0**



chain code 1,0,0,0,1,2,2,1,2,4,3,4,4,4,6,6,6,6,5,6,6

predictive CC: **7,0,0,1,1,0,7,1,2,7,1,0,0,2,0,0,0,7,1,0**

Motivation for Compression and Reformatting

- reduction in storage-memory demand
  - accelerated data transfer
  - reduced storage demand when archiving the information
  - limited resources on the target device, e.g. reformatting to match the target display resolution
- accelerated processing
  - multi-resolution approaches (*c.f. filter pyramid*) utilized to speed-up very time consuming segmentation, filtering and registration approaches by reducing size of input data for first rough approximation
- normalization due to resampling and rescale
  - in case of registration or building up a statistical a priori model, mismatches in extent and datatype need to be harmonized

Compression - Overview

- schematic classification
  - ***lossless***: original signal / information can be reconstructed, i.e. decoded. E.g. ZIP, runlength coding,...
  - ***lossy***: applied compression is irreversible, e.g. JPSEG, quantization,...
  - utilizing lossy compression strategies, generally much higher compression rates are achievable
- compression strategies
  - ***spatial extent***:
    - reducing size (resampling)
  - ***scalar range***:
    - reducing scalar-range (data type), e.g. by applying quantization
    - code-transformations exploiting low levels of entropy
  - ***information content***:
    - exploiting redundancy
    - removing irrelevant frequencies, lossy compression

Types of Redundancy in Image Data

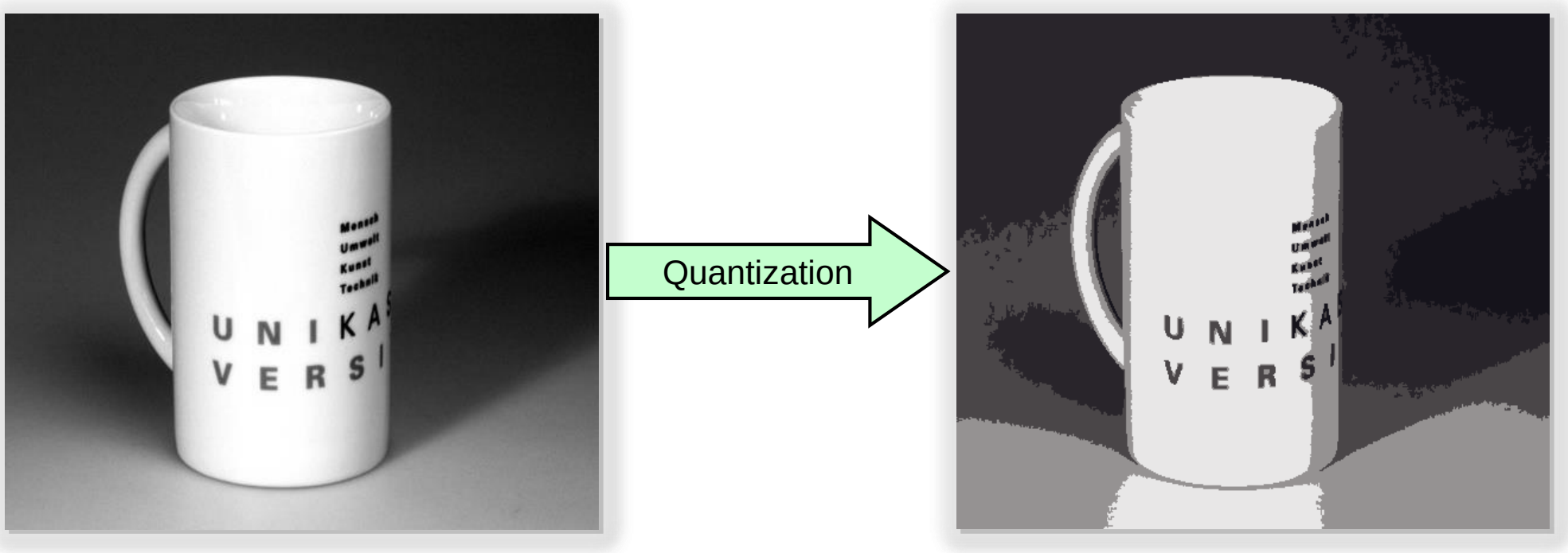
- Coding Redundancy
  - the most dominant symbols (letters, scalar values,...) should be represented by the shortest encoding, e.g. applying *Huffmann Coding* or Lempl-Ziv (c.f. ZIP)
- Inter-Pixel-Redundancy
  - pixel values are not randomly distributed. Instead, scalar values highly depend on the local neighbour values, e.g. *runlength coding, bitplane-coding or e.g. DCT* (Discrete Cosine Transformation) for JPEG
- Nature of Human Sensing, psycho-visual redundancy
  - the human visual system cannot discriminate an infinite number of colours.
  - w.r.t. chromaticity, a human can discriminate around *200* shades, mainly in the green section, c.f. CIELAB colour space.
  - besides, a human can discriminate around *60-80* shades of grey.
  - Thus, quantization often will not lead to perceptible loss in quality

Redundancy - Basic Definitions

- ***data***: raw, “unorganized” facts that may contain different levels of ***information***
- data can be seen as container of symbols to hold information, e.g. letters “d”,”o”,”g” or <100><111><103> in decimal-ASCII vs. semantic meaning of *dog* as animal.
- compression-rate of original dataset n_1 and compressed dataset n_2 holding the “same” information calculated as $CR = \frac{n_1}{n_2}$ with n_1, n_2 as data units, e.g. bits.
- the relative ***redundancy*** is calculated as $RD = 1 - \frac{1}{CR}$
  - with $n_1 \cong n_2 : RD \to 0$,$n_2 \ll n_1 : RD \to 1$
- ***entropy***: lack of predictability, amount of information within data
  - in case of low entropy, generally higher compression rates are achieved
  - *Shannon Entropy*: calculated base on the occurrence probability of the coding symbols, e.g. scalar value distribution of input image histogram as $H = -\sum_{i=1}^{n} p_i \cdot \log p_i$
- besides entropy, ***homogeneity***, i.e. similarity/ordering of local symbols, relevant for many compression strategies too

Quantisation and Resampling

- due to psychovisual redundancy, the scalar range often can be reduced to still recognize the image scene, e.g. $[0; 255] \rightarrow [0; 4]$ as extreme example



- furthermore, reduction in size can lead to significant compression rates.
  - In the medical domain, reducing each dimension of a tomographic 3D dataset leads to a data reduction of 87.5%.

Runlength Coding

- efficient compression strategy in case of low entropy and high homogeneity
  - binary encoded data: instead of storing bit-by-bit, the length of “0” and “1”-sequences is encoded
  - e.g. input data $a = 1111000000001111001111111100000011111000$
    - length of 40 symbols, thus 40bit
    - with runlength coding and “0” defined as starting bit: $b = 0\ 4\ 8\ 4\ 2\ 8\ 6\ 5\ 3$ thereby reducing to 9 symbols. If the sequence-length datatype exemplarily is encoded with 4 bit, occurrences of [0..15] can be represented. The data amount is reduced to $4 * 9 = 36$ leading to a compression rate $CR = 1.\dot{1}$.
    - but what if there are longer sequences of the same symbol in the input signal? Represent the maximum number, encode an “0” as intermediate alternating step and continue with the rest.
  - if the symbol set has more than two symbols, runlength coding might get inefficient.
    - instead of pre-determined symbol-altering order the symbol itself needs to be encoded prior to its occurrence number, e.g. $a = 111111112222222222233333333331111111 \rightarrow b = \mathbf{1}|8|\mathbf{2}|12|\mathbf{3}|10|\mathbf{1}|7$
  - used for Microsoft bitmap, Fax, JPEG compression,...

Predictive Coding

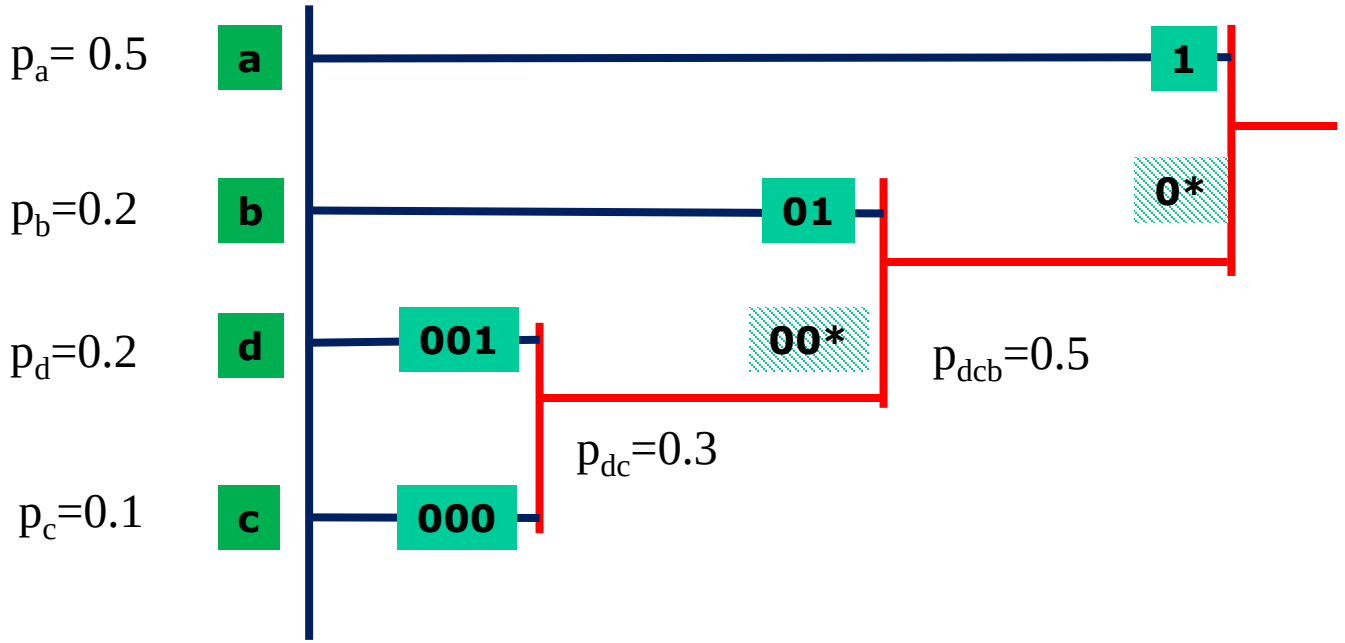
- Homogeneity of symbol sequence can be slightly increased by applying *predictive coding*.
- thereby, the local gradients, i.e. intensity difference comparing in local neighbourhood, are utilized, thus significantly reducing the scalar range
  - the smaller gradients/deltas can be represented in a more efficient way
  - runlength coding applied on the predictive sequence will lead to higher compression rates compared to applying on the original symbol sequence
- example:
  - ***a***=1111111222222222223333333331111111
  - ***b***=10000000100000000001000000000-2000000
- based on first symbol and the predictive sequence, the original sequence ***a*** can be reconstructed

Huffmann Coding

- compression rate achievable by Huffmann Coding highly depends on entropy of the symbol sequence
- thereby, the binary encoding of the symbols is compressed
- the resulting coding features a very compact binary encoding for the most relevant symbols and a significant overhead for the symbols occurring less often.
- Huffmann tree calculated based on the symbol probabilities p_i.
  - leaves are sorted according to increasing probability
  - Huffmann tree utilized for both, encoding and decoding
  - no additional delimiters required as restructured codes sill guarantee unambiguous parsing
- example: symbol set {a,b,c,d} generally encoded by 2bit
  - a=00, b=01, c=10, d=11
  - sequences often show dominant symbols (c.f. voxels in German language compared to x,y,z)
  - e.g. sequence s_1= abaaabbddbcdcdaaaaaa, 20symbols, 40bit.
  - probabilities $p_a = 0.5$, $p_b = 0.2$, $p_c = 0.1$, $p_d = 0.2$

Huffmann Coding cont'd

- Huffman Tree is built up bottom-up, while the coding is assigned top-down
- a=1, b=01, d=001, c=000, binary encoded: 10*1 + 4 * 2 + 2 * 3 + 4 * 3 = **36bit**
- CR to be deduced by mean encoding length $L = \sum_r l(r) \cdot p(r)$
  - for the example: L= 1 * 0.5 + 2 * 0.2 + 3 * 0.2 + 3 * 0.1 = **1.8**, compared to uncompressed 2 bit



Bit-plane Coding

- specific compression strategy for images with byte-based pixel datatypes
- a scalar value in range [0;255] can be represented as polynomial of 8 bits with $s = c_7 \cdot 2^7 + c_6 \cdot 2^6 + \cdots + c_1 \cdot 2^1 + c_0 \cdot 2^0$
  - thereby, all of the binary coefficients $c_0..c_7$ can be used to build up 8 binary image planes (*bit-plane decomposition for gray-scale images*)
  - while at a particular pixel position c_7 determines if the scalar value is above or below 128, the influence of c_0 is ± 1 at most.
  - thus, the lower bit showing marginal influence can be compressed or used for hidden data encoding (steganography)
  - in contrast, the higher bits are expected to show increased homogeneity, thus predestined for applying runlength coding
  - proper compression strategy: *"ignore" the lower bit-planes and compress the higher bit-planes*

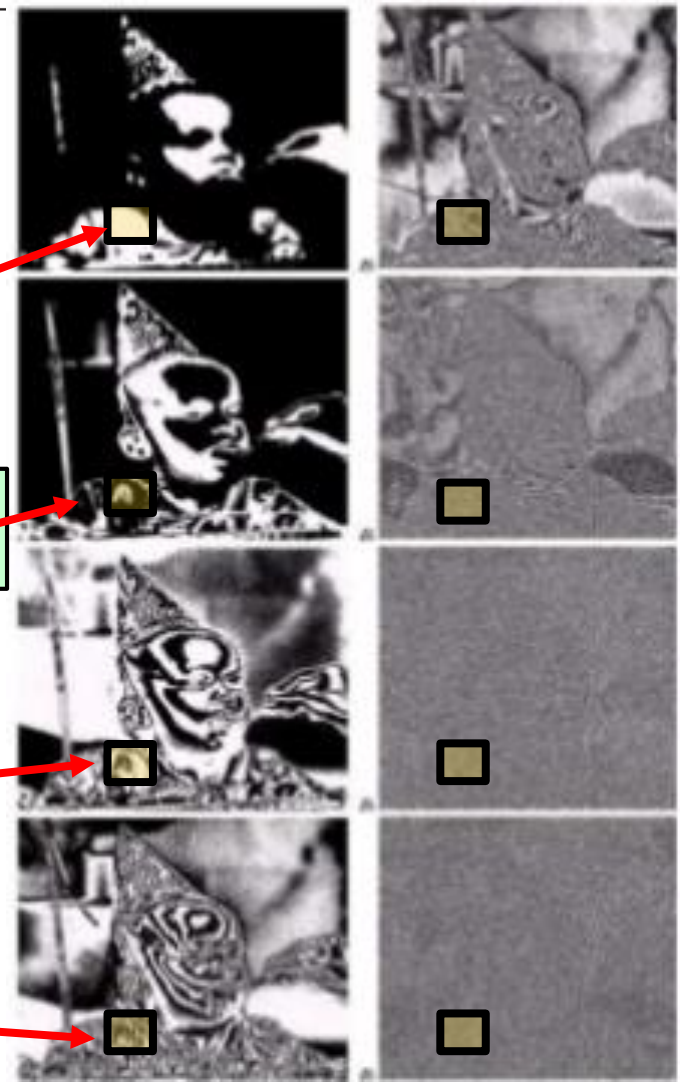
Bit-plane Coding cont'd

- example for bit-plane-decomposition

original image

$I(100,120) = 210 = 128 + 64 + 16 + 2$

bit-planes $\mathbf{C_7}$ (top-left) to $\mathbf{C_0}$ (bottom-right)



1 (128)	**1** (64)	0 (32)	**1** (16)	0 (8)	0 (4)	**1** (2)	0 (1)
c_7	c_6	c_5	c_4	c_3	c_2	c_1	c_0

Lempel-Ziv-Welch Compression (ZIP)

- lossless, utilized for many image formats (TIFF, GIF,...)
- while encoding, a dictionary of repetitive symbol sequences is constructed allowing efficient indexing
- size of dictionary is pre-defined, e.g. 4,096 (2^{12}) entries and [0;255] for terminal symbols
  - the dictionary covers sequences of length $n \geq 2$ only.
  - for each symbol(sequence) processing step, the dictionary MUST get an additional entry
  - parsing covers current and next *1-n* symbol(s)
  - encoded sequence **implicitly contains the dictionary** – can be restored while decoding

Lempel-Ziv-Welch Compression (ZIP) cont'd

UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA
OBERÖSTERREICH

- example: sequence s=*ababbbaabbcdcdcdcabbc* [21 symbols]

current symbol	next symbol	in dictionary? –> encoded symbol(s)	dictionary
a	b	N -> **a**	ab <256>
b	a	N -> **b**	ba <257>
a	b	Y, abb: N -> **256**	abb <258>
b	b	N -> **b**	bb <259>
b	a	Y, baa:N -> **257**	baa <260>
a	b	Y, abb Y, abbc:N -> **258**	abbc <261>
c	d	N -> **c**	cd <262>
d	c	N -> **d**	dc <263>
c	d	Y, cdc:N -> **262**	cdc <264>
c	d	Y, cdc:Y, cdca:N -> **264**	cdca <265>
a	b	Y, abb:Y, abbc:Y -> **261**	---

  - result: **<97><98><256><98><257><258><99><100><262><264><261>** [11 symbols, CR≈2]

Outlook: Compression for Classification

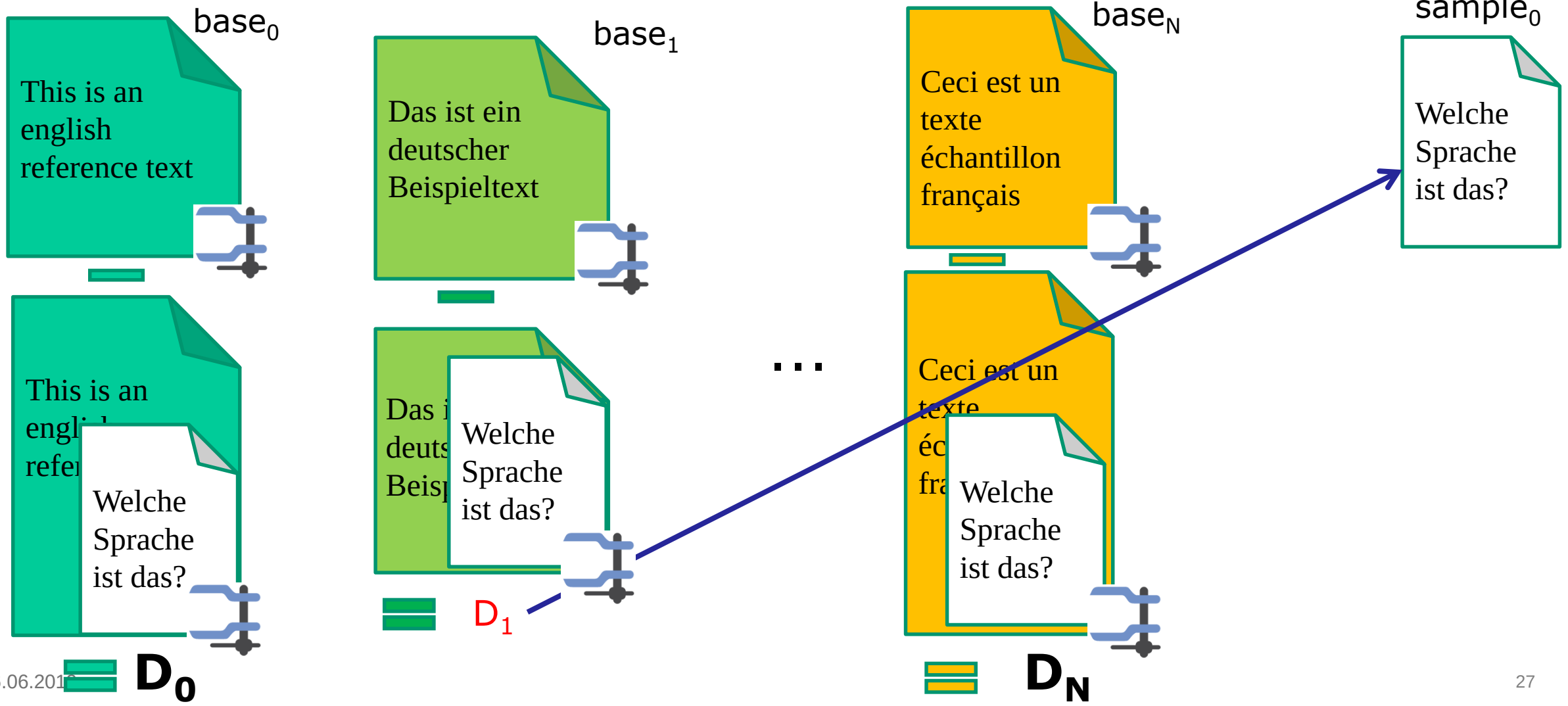
- Dictionary-based compression approaches like gzip [Deutsch 1996] or common LZW-Compression [Ziv and Lempel 1978; Welch 1984] are applicable to various classification domains:
  - classification of **text** (identifying the language or text similarities)
  - classification of **images** (assigning images to categories) [Heidemann and Ritter, 2008]
  - classification of **language** signals (identifying spoken language and the speaker based on phonemes) [Zwettler and Horn, 2008]
- basic concept: a reference signal ref_i or a set of reference signals is provided for each of the classes C_i that need to be discriminated.
- The classification process itself is based on evaluating the maximum achievable compression rate of a sample signal $sample_j$ successively concatenated and compressed with all of the provided reference signals.
- Thus, the sample is classified according best match with the particular classes compression dictionaries. The more sub-sequences of $sample_j$ are already contained in the dictionary, the higher the achievable compression rate becomes.

Outlook: Compression for Classification cont'd

- utilization of compression SW to compress each reference dataset ref_i together with the unknown signal $sample_j$ to be classified one after another.
- the achievable compression delta Δ of compressed ref_i as $comp(ref_i)$ compared to $comp(ref_i \cup sample_j)$ is evaluated per class C_i of n classes to discriminate.
  - $c = arg\,min_{i=1..n} |comp(refi) - comp(refi \cup sig_j)|$
- the coding of the data is thereby of high relevance:
  - the input data must be uncompressed
  - the sample and reference signals must show the same character coding, e.g. ASCII or 8bit unsigned for gray-value images
  - signal size with respect to the dictionary capacity and fill level

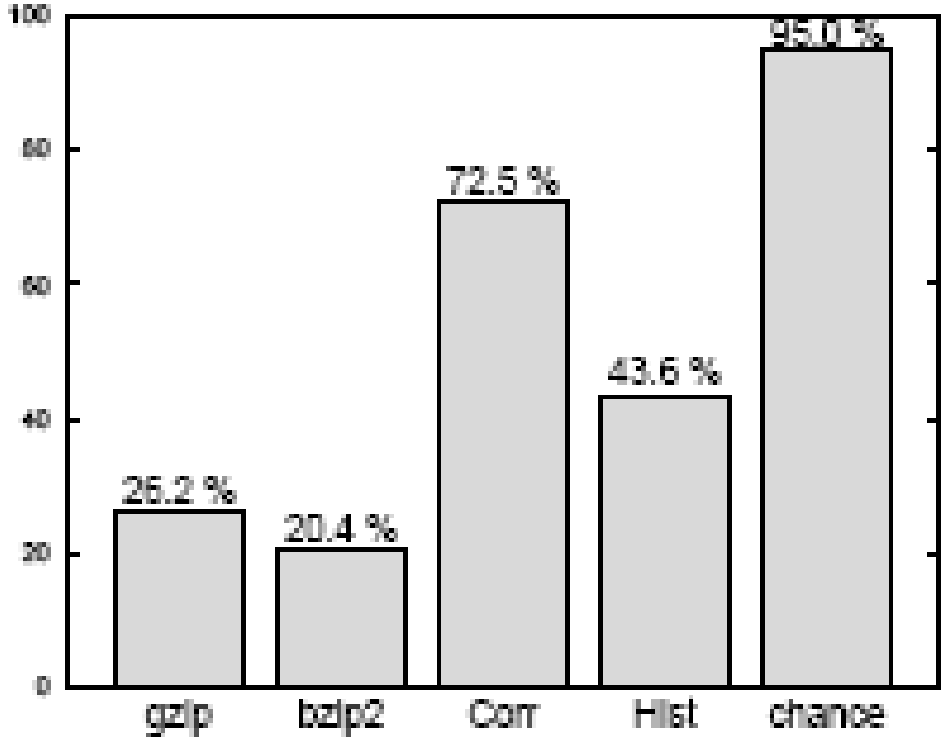
Outlook: Compression for Classification

- In the following, the classification approach is illustrated for plain text signals



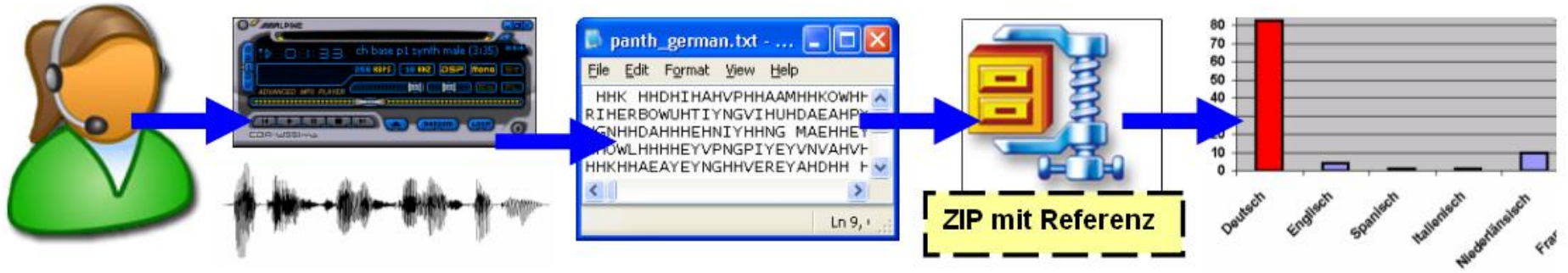
Outlook: Compression for Classification

- classification of image data, n=20 classes



Outlook: Compression for Classification

- compression of spoken audio files, phoneme encoded speech

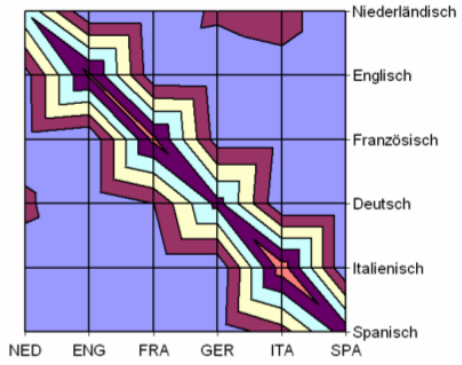


	NED		ENG		FRA		GER		ITA		SPA	
Samples	*R*	*K*	*R*	*K*	*R*	*K*	*R*	*K*	*R*	*K*	*R*	*K*
Niederländisch	**1**	**58**	4	1	5	0	3	19	2	22	5	0
Englisch	2	9	**1**	**77**	4	4	5	1	2	9	6	0
Französisch	2	7	6	0	**1**	**78**	2	7	2	7	5	2
Deutsch	2	18	4	1	3	15	**1**	**65**	4	1	6	0
Italienisch	2	8	3	4	4	2	4	2	**1**	**83**	6	0
Spanisch	6	0	5	1	4	2	3	5	2	23	**1**	**69**

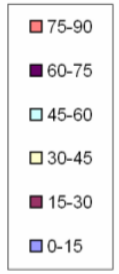
(a)

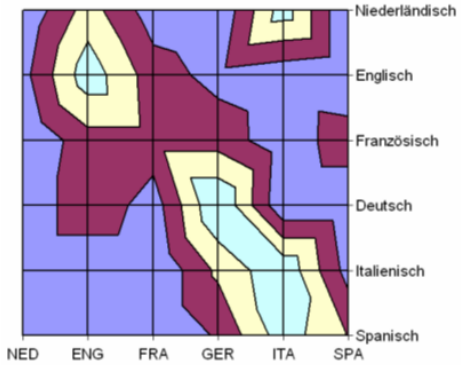
	NED		ENG		FRA		GER		ITA		SPA	
Samples	*R*	*K*	*R*	*K*	*R*	*K*	*R*	*K*	*R*	*K*	*R*	*K*
Niederländisch	**4**	**2**	2	37	3	7	4	2	**1**	**53**	6	0
Englisch	3	10	**1**	**54**	2	22	3	10	5	4	6	0
Französisch	6	0	1	24	**1**	**24**	1	24	5	5	1	24
Deutsch	6	0	2	28	3	7	**1**	**60**	4	2	4	2
Italienisch	4	0	4	0	4	0	2	33	**1**	**57**	3	9
Spanisch	4	0	4	0	4	0	3	17	**1**	**53**	**2**	**29**

(b)



(c)





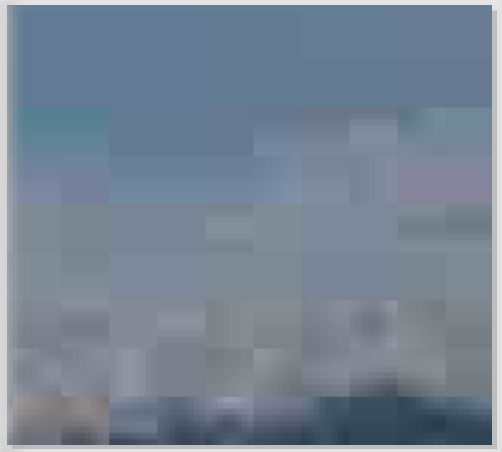
(d)

(a)(c) synthesized speech
(b)(d) human speaker (non-native)

JPEG Compression

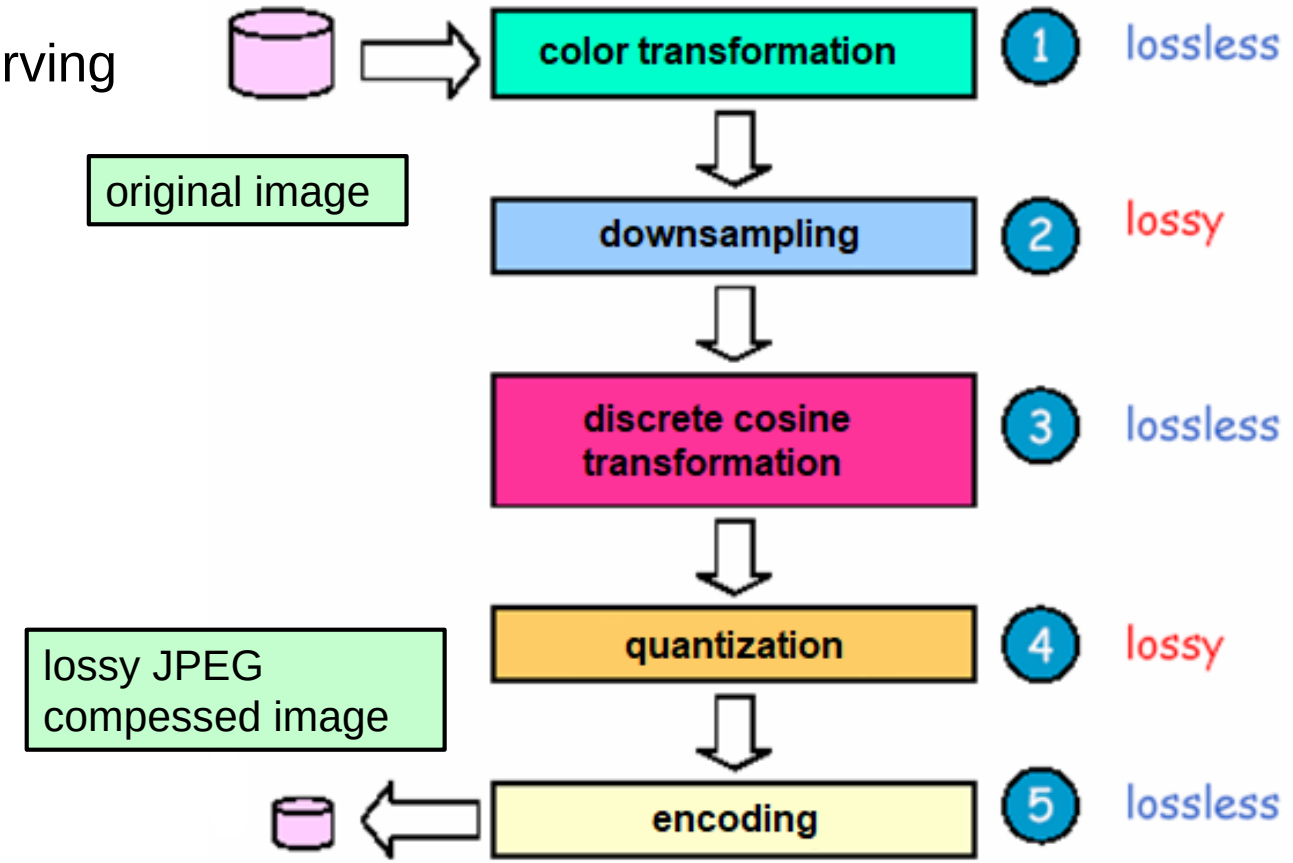
- JPEG (ISO/IEC 10918-1 from **J**oint **P**hotographics **E**xperts **G**roup) allows compression rates of around *20*.
  - while inside 8x8 block the pixel colours look blurred, the borders of the 8x8 blocks are clearly visible in lossy compressed JPEG

original image with 2000KB
compressed down to 111KB



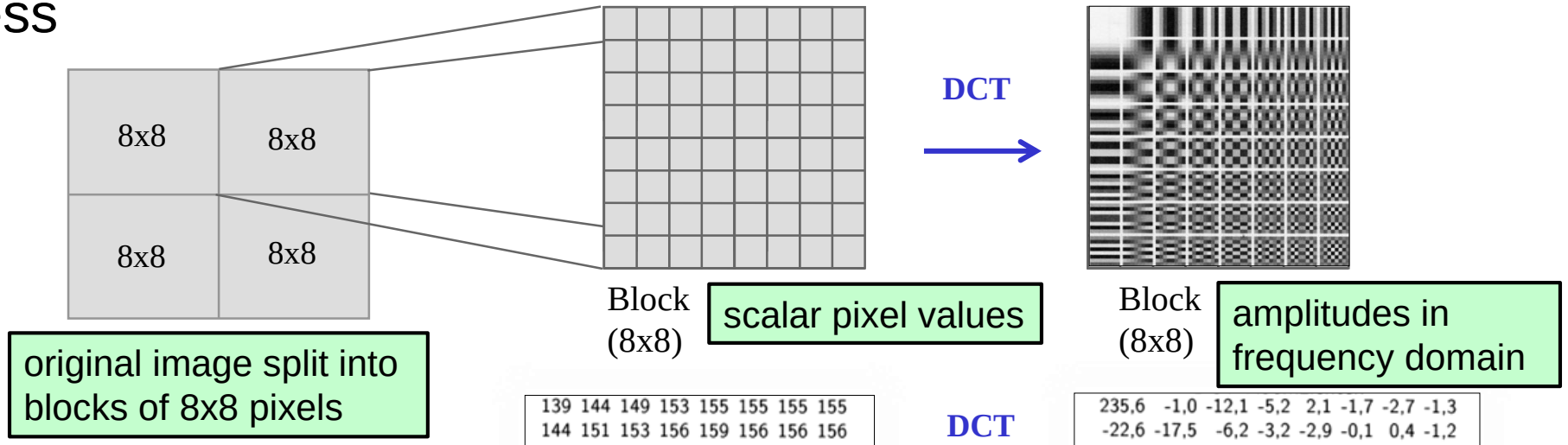
JPEG Compression cont'd

- several compression strategies are applied in the context of JPEG image format:
  - colour transformation from RGB ➔ YUV
  - down-sampling of U,V channel while conserving luminance (Y)
  - transformation to Frequency domain utilizing DCT (discrete cosine transformation)
    and applying quantization
  - Zigg-Zagg order runlength coding



JPEG Compression cont'd

- DCT applied to blocks of 8x8 pixels, interpreting these ROIs as separate images to compress



JPEG Compression cont'd

- amplitudes still allow lossless compression
- to get integer values for subsequent encoding, all amplitude values are divided by coefficients of a quantization table
  - quantization table is stored in the JPEG header and steering the level of compression / quality conservation

235,6	-1,0	-12,1	-5,2	2,1	-1,7	-2,7	-1,3
-22,6	-17,5	-6,2	-3,2	-2,9	-0,1	0,4	-1,2
-10,9	-9,3	-1,6	1,5	0,2	-0,9	-0,6	-0,1
-7,1	-1,9	0,2	1,5	0,9	-0,1	0,0	0,3
-0,6	-0,8	1,5	1,6	-0,1	-0,7	0,6	1,3
-1,8	-0,2	1,6	-0,3	-0,8	1,5	1,0	-1,0
-1,3	-0,4	-0,3	-1,5	-0,5	1,7	1,1	-0,8
-2,6	1,6	-3,8	-1,8	1,9	1,2	-0,6	-0,4

DCT coefficients (amplitudes)

/

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

quantization table

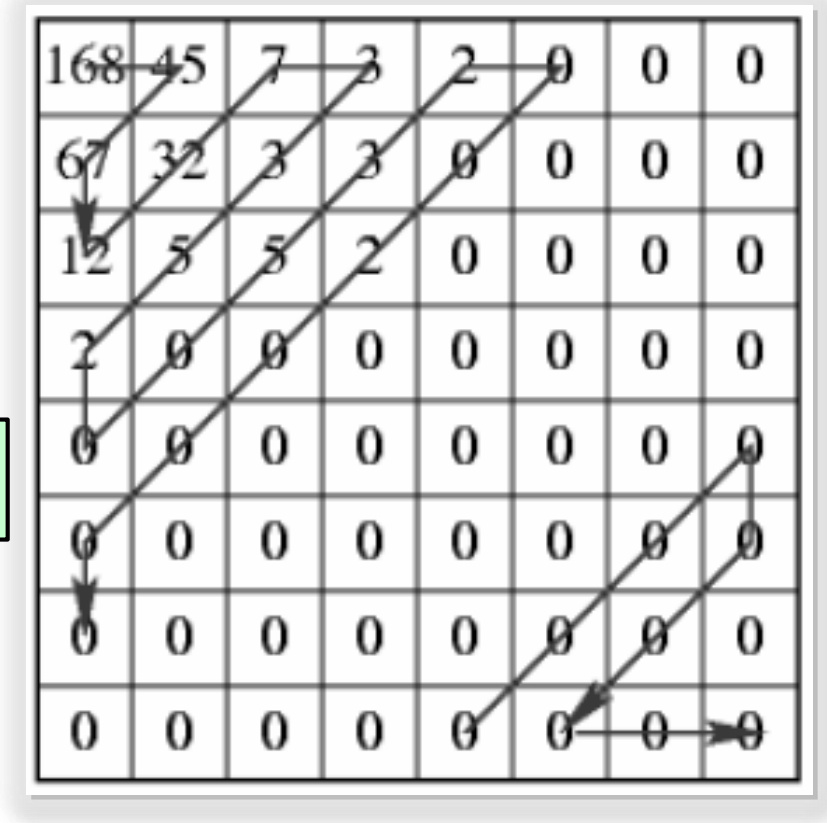
=

15	0	-1	0	0	0	0	0
-2	-1	0	0	0	0	0	0
-1	-1	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

quantized coefficients

JPEG Compression cont'd

- after performing the DCT and quantization of the coefficients, entropy encoding is applicable utilizing runlength coding
- to feature a high level of homogeneity, *Zigg-Zagg* order for processing the coefficients is applied, thereby starting in the top left corner and traversing the frequencies in diagonal direction,
  i.e direction of increasing frequencies.



Zigg-Zagg processing order after quantization of the coefficients

Bibliography

- M. Sonka, V. Hlavac, and R. Boyle – "*Image Processing: Analysis and Machine Vision, London*, Chapman & Hall Computing", 1993.
- G. Heidemann, and H. Ritter – "On the Contribution of Compression to Visual Pattern Recognition", In: Proc. of VISAPP 2008 - 3rd International Conference on Computer Vision Theory and Applications, no. 2, 2008, pp. 83-89.
- J. Ziv, and A. Lempel – "Compression of individual sequences via variable-rate coding", In: Transactions on Information Theory 24(5), 1978, p. 530.
- G. Zwettler, and G. Horn – "Phoneme Encoded Audio Stream Language Identification Based on Generic Compression Strategy", Technical Report, Upper Austria University of Applied Sciences, 2008.

UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA

Image Processing

Registration

DI(FH) Dr. Gerald Adam Zwettler, MSc, gerald.zwettler@fh-hagenberg.at
Lector of Computer Graphics, Image Processing and Computer Vision

Registration - Terminology

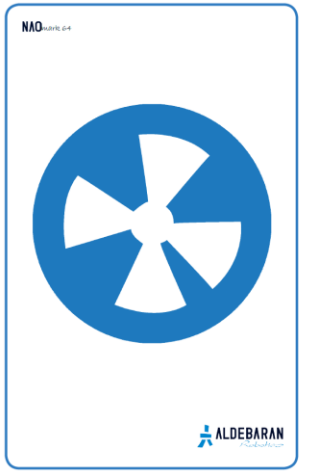
- registration refers to an alignment process for two or more images of the same scene to ensure matching content, objects or structures.
- the static image or pattern is thereby denoted as **reference image A**, while the other **moving/object image(s) B** are transformed in an optimization process to achieve a good match.
- thus, registration is a process to optimize an affine transformation $T(B) = B'$ that minimizes the error between the reference and the object image $diff(A, B')$.
  - registration process: $\hat{T} = \underset{\boldsymbol{T}}{\operatorname{argmin}}[diff(A, T(B))]$
- relevant affine transformations:
  - *translation*
  - *rotation*
  - *scale*

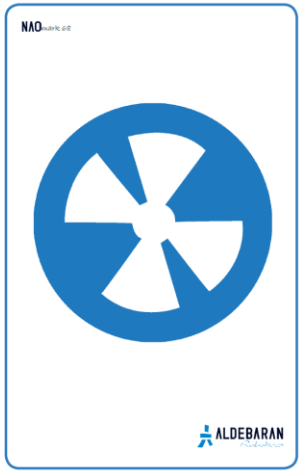
Registration Overview

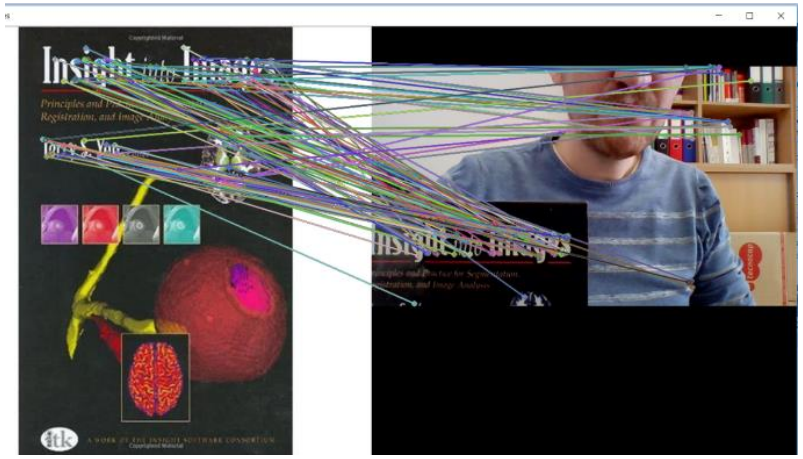
- application domains:
  - detection, tracking and alignment:
    - detection of objects/structures in images, pattern matching
    - object tracking, c.f. surveillance videos
    - building up a statistical shape model (SSM) utilizing Procrustes alignment
  - multimodal image analysis:
    - e.g. combining colour and grayscale images
    - sensor fusion and visualization
    - medical domain: multi-modal image analysis, e.g. combining MR/CT and SPECT/PET
  - analysis of deformations and transformations:
    - deriving the spatial misalignment of images or signals
    - e.g. for shape comparison, camera calibration, SLAM algorithm, stereo-reconstruction or image stitching

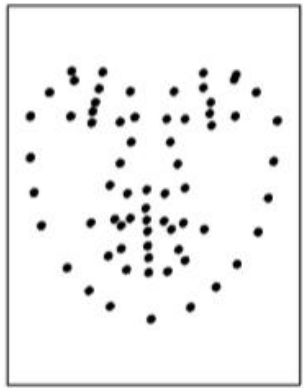
Registration Overview cont’d

- detection, tracking and alignment:



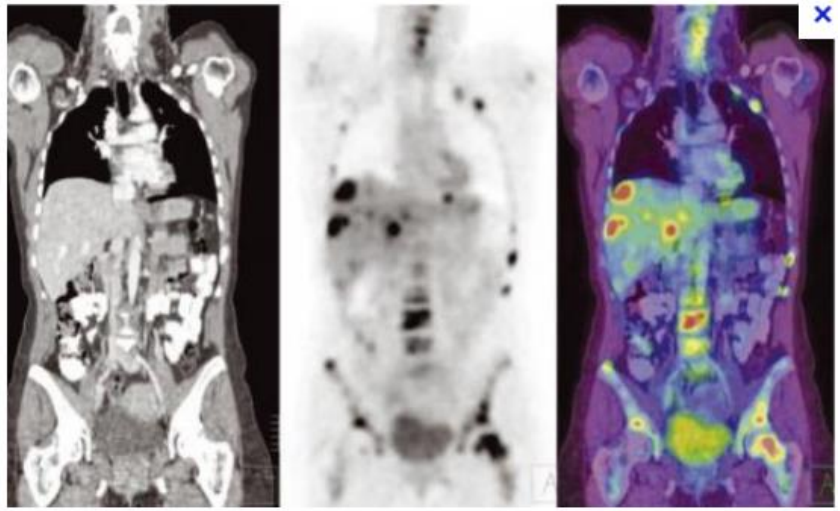


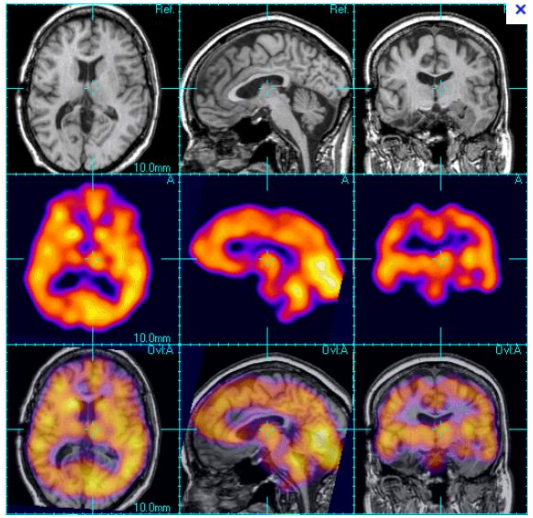


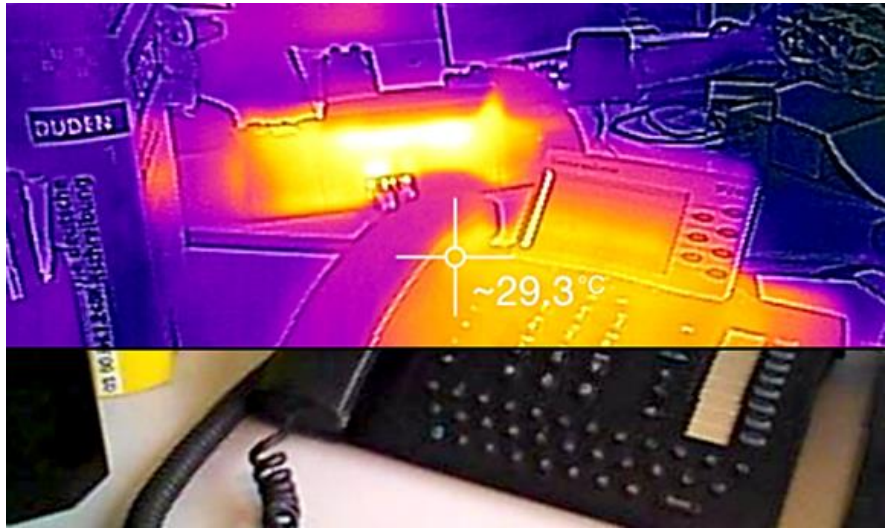


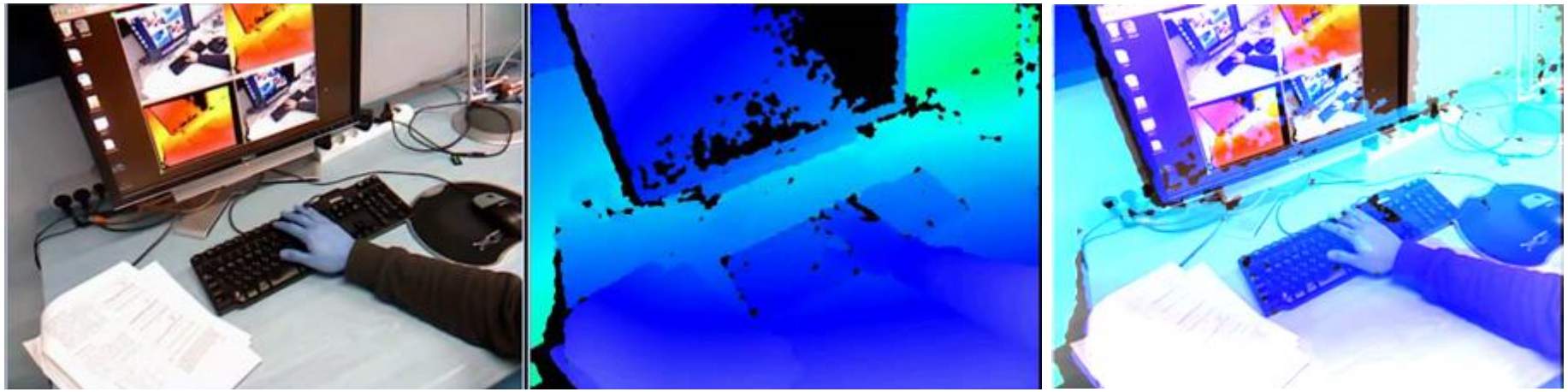
Registration Overview cont'd

- multimodal image analysis:



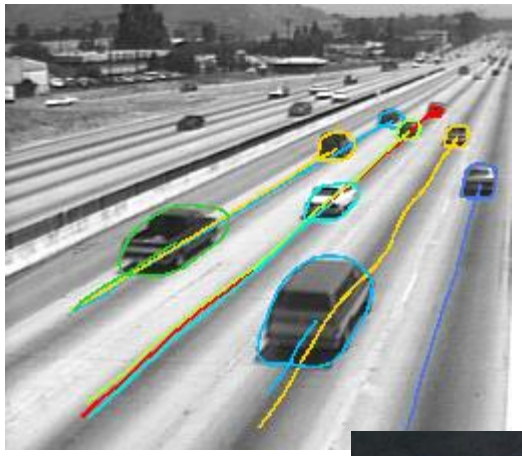


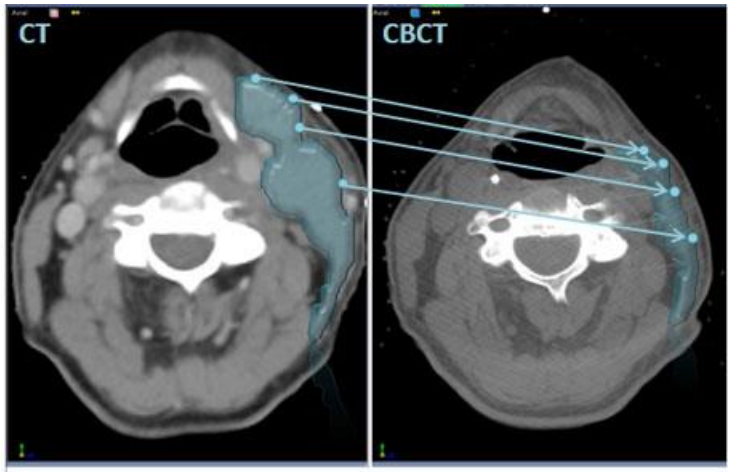




Registration Overview cont'd

- analysis of deformations and transformations:





Affine Transformations

- affine transformations, denoted as rigid body transforms, are projections from affine space A to affine space B
  - parallelism is conserved
  - aspect-ratio is conserved
- rigid body transformation:
  - **translation** and **rotation** as key operations
  - **scale** only required in case of mismatching proportions
  - **shearing** required in case of perspective distortions
- due to affine transformations, validity of the pixel positions needs to be ensured
  - padding utilizing neutral scalar values as adequate strategy

Affine Transformations cont'd

- translation (x,y,z,...)
  - adding static offset in each direction

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

translation $T = \begin{bmatrix} 45 \\ -15 \end{bmatrix}$

Affine Transformations cont'd

- rotation
  - one rotation parameter for 2D, at least three rotation parameters for 3D
  - attention: rotation should be performed for image coordinate system with origin in the image mid

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad \begin{aligned} x' &= x \cdot \cos(\theta) + y \cdot \sin(\theta) \\ y' &= -x \cdot \sin(\theta) + y \cdot \cos(\theta) \end{aligned}$$

rotation R = 20°

Affine Transformations cont'd

- scaling
  - generally only the image content is thereby adjusted
  - scaling should be performed with origin in image center
  - adapting the image size itself is denoted as resampling and performed in advance
  - scale factor to be defined per dimension

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

scale $s_x = s_y = 0.5$

Affine Transformations cont'd

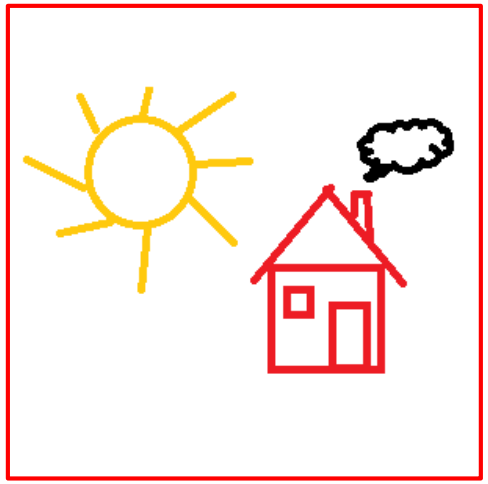
- shearing
  - rotating one axis while keeping the other axis constant

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} 1 & s \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}, x' = x + s * y$$

shear $s = 0.1$

Error Metrics

- to solve registration as optimization task, quality, i.e. the level of congruence of A and B', needs to be assessed in a quantitative way
- thereby, the quantitative error metric should correlate with the proximity to the theoretical global best transformation solution.

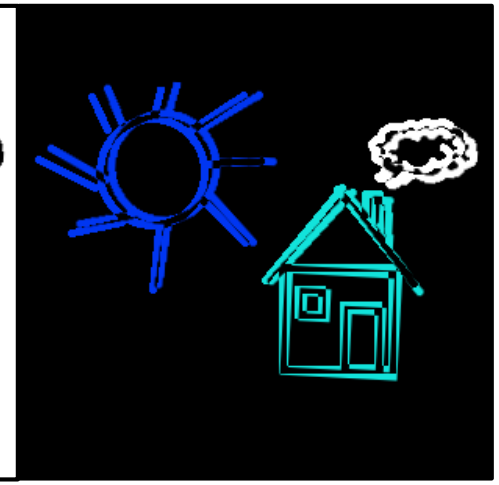


A

B_1, rot 15, T_x=15

Err_{mean}=17.21

B_2, rot 3, T_x=5



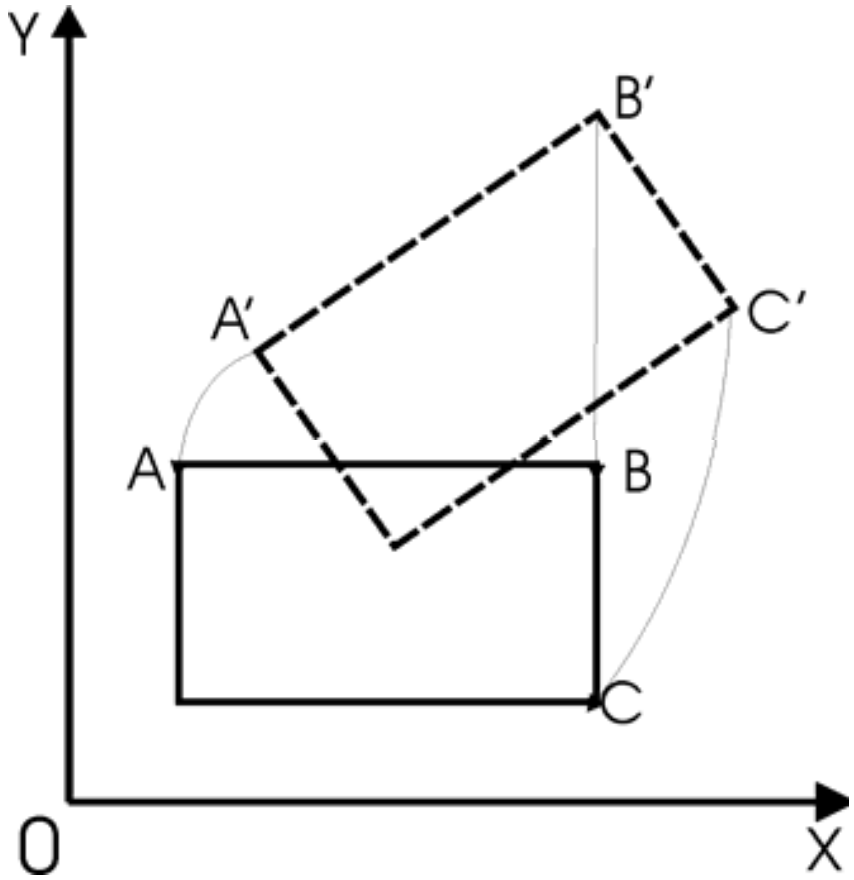
Err_{mean}=13.80

Error Metrics cont'd

UNIVERSITY OF APPLIED SCIENCES UPPER AUSTRIA
OBERÖSTERREICH

- **overview of applicable metrics**
  - difference in intensity profile as **SSE** (sum of squared error)
  - **distance** of uncorrelated discrete points along pre-segmented object's borders:
    - *ICP* (iterative closes points)
    - Chamfer Matching utilizing an *Euclidean Distance Map*
  - matching of **corresponding landmarks**
  - cross-intensity correlation calculated as **mutual information**
  - **visual inspection** for manual registration, overlay CTF (colour transfer function)

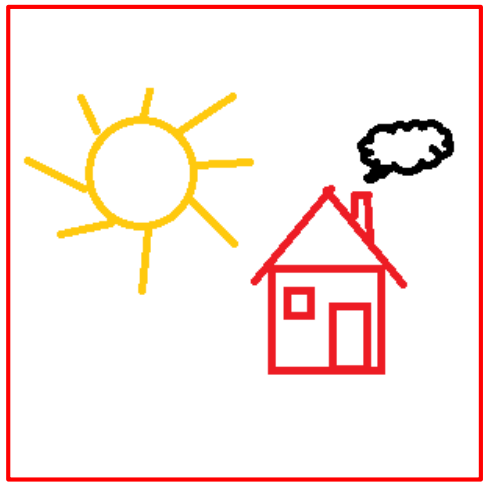
Corresponding Landmarks



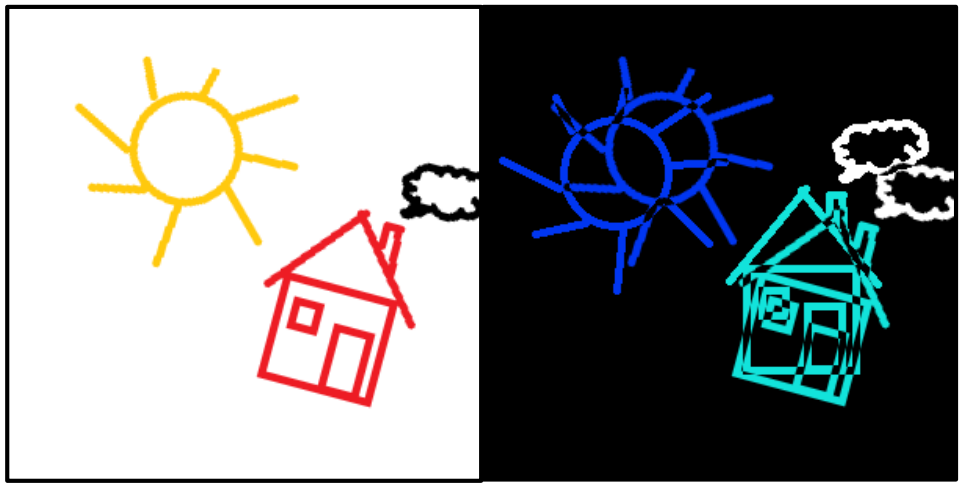
- given a sufficient number of precise corresponding landmarks in both images, the transformation can be calculated in a numeric way, e.g. utilizing SVP (*singular value decomposition*) or Falk-scheme for equation solving of $Ax = b$.
  - for 2D registration, *3* landmarks are sufficient if building up a unique shape (not on single line, no symmetry,...)
  - for 3D registration, around *5-6* landmarks are required at least (not all on line, plane, no symmetry,...)
- utilizing numeric approaches the distance between the corresponding point sets gets minimized
- hard to derive these landmarks in an automated way. Manually placed landmarks rather expected to lead to inaccurate results

SSE calculated on intensity profile

- for two images A and B' the squared difference at all pixel positions is summed up as SSE with $SSE(A,B) = \sum_{x=1}^{width} \sum_{y=1}^{height} (A(x,y) - B(x,y))^2$

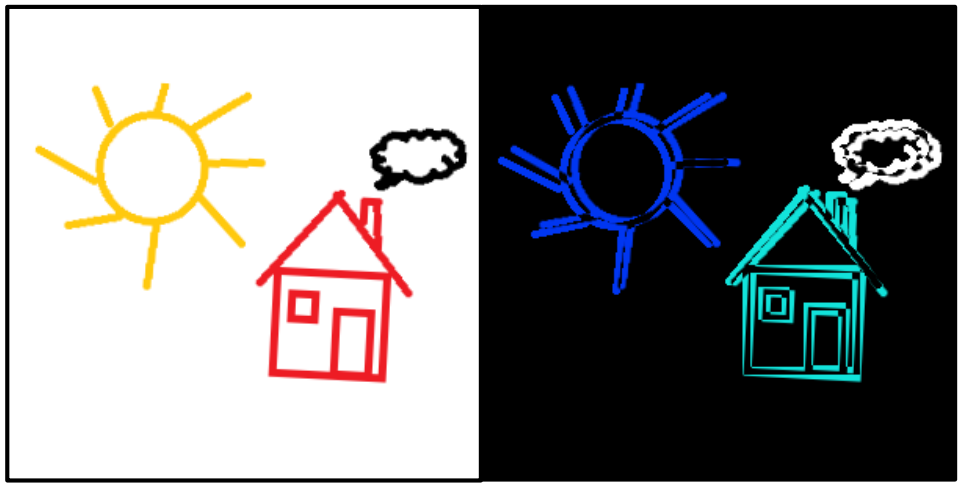


A



B_1, rot 15, T_x=15

Err_{SSE}=2.45*10^8



B_2, rot 3, T_x=5

Err_{SSE}=1.98*10^8

Distance between uncorrelated discrete points

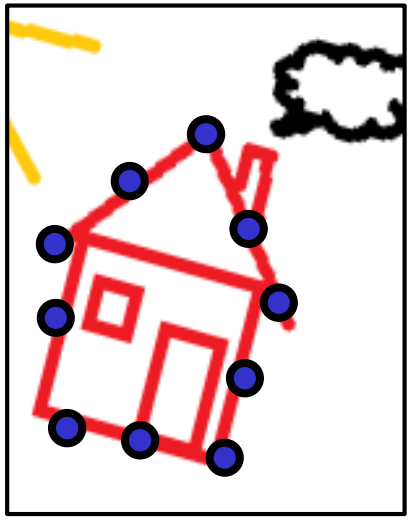
- both images A and B' are (partially or fully) segmented to subsequently extract borders.
- To describe the shape in a characteristic way, an equally distributed discrete number of points along the borders is extracted, denoted as $\mathbf{P_A}$ and $\mathbf{P_{B'}}$, with generally $size(\boldsymbol{P_A}) \neq size(\boldsymbol{P_{B'}})$.
  - the size of the border point sets thereby balances between robustness/accuracy and performance
- **ICP algorithm**: as no point correlation is given, the distance error per point of $\mathbf{P_A}$ is iteratively calculated checking all distances permutations with $\mathbf{P_{B'}}$. Thereby $size(\boldsymbol{P_A}) \cdot size(\boldsymbol{P_{B'}})$ distances need to be evaluated in total.
- Overall error is calculated as sum of squared distances with

$$Err_{ICP} = \sum_{i=1}^{size(\boldsymbol{P_A})} \left(MIN_{j=1}^{size(\boldsymbol{P_{B'}})} |\boldsymbol{P_{Ai}} - \boldsymbol{P_{B'j}}| \right)^2$$

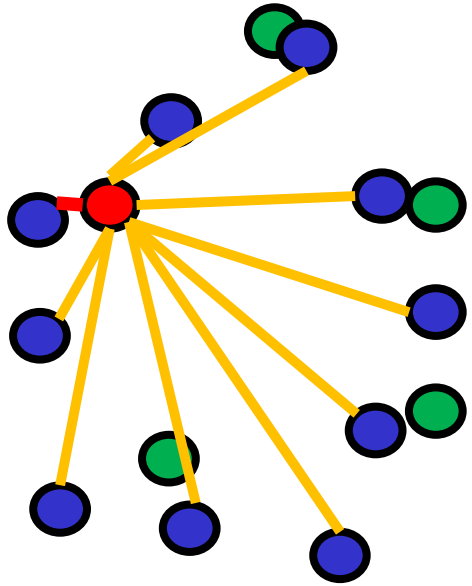
Distance between uncorrelated discrete points

- ICP Example: $size(\boldsymbol{P}_A) = 5, size(\boldsymbol{P}_{B'}) = 10$. In total $5 * 10 = 50$ permutations to be checked for summing up the 5 squared distances

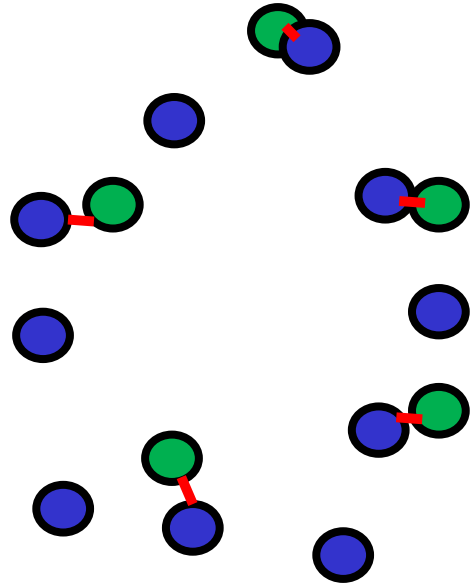
A



B‘



for each point in $\boldsymbol{P}_A$ (marked red), the closest distance to any point in $\boldsymbol{P}_{B'}$ is calculated



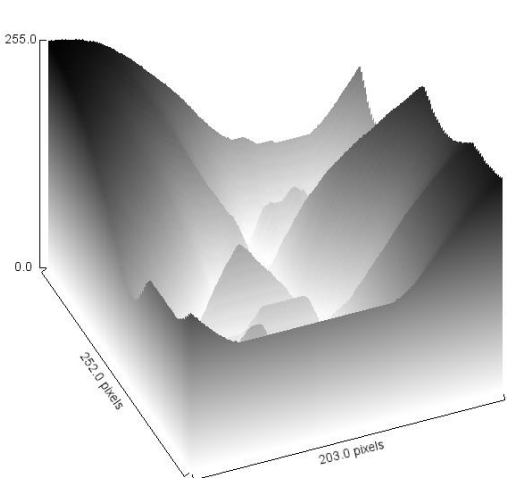
total error as sum of $size(\boldsymbol{P}_A)$ squared smallest distances between $\boldsymbol{P}_A$ and $\boldsymbol{P}_{B'}$

Distance between uncorrelated discrete points

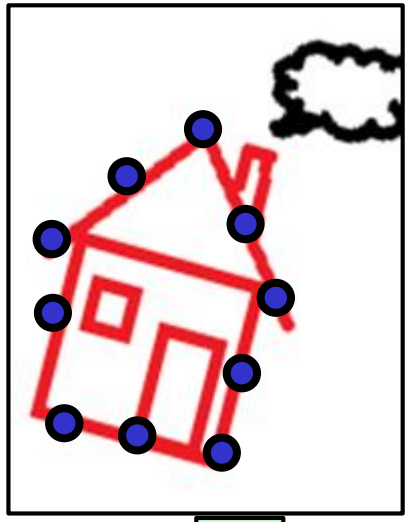
- Chamfer Matching: to speed-up ICP, for the static reference image **A**, an Euclidean Distance Map $\mathbf{DM_A}$ can be calculated once, holding in every pixel the distance to the closest border element for ad hoc evaluation.
  - thus, runtime complexity can be reduced to $O(n)$ for direct indexing compared to $O(n * m)$ for ICP.

A

$\mathbf{DM_A}$



profile of $\mathbf{DM_A}$



B‘

sum of squared DM at B-point index positions

Calculation of Distance Map

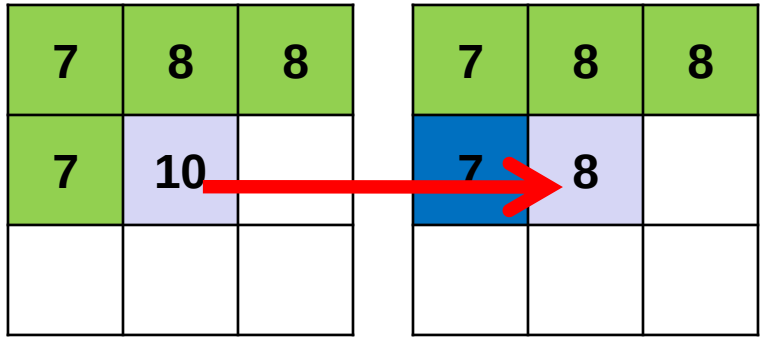
- calculation of the distance map as iterative optimization problem
- initially, all border elements are assigned a distance value of **0**, while all other pixels are assigned an undefined distance of ∞.
  - iterating over the image, pixels might be updated, i.e. assigning a lower value, by checking the distances currently assigned to the direct neighbours.
  - weight of *p* is updated, if $dist(q) + dist(p,q) \leq dist(p)$ with *p,q* in N_8.
  - for distance map calculation, the neighbourhood matrix contains 4 positions only, leading to $dist'(p) = min\left(dist(p), \bigvee_{i=1}^{4}(dist(RU_i) + dist(p, RU_i))\right)$

RU_1	RU_2	RU_3
RU_4	P	

adjacency for run from top left to bottom right (**RU**)

2	1	2
1	P	1
2	1	2

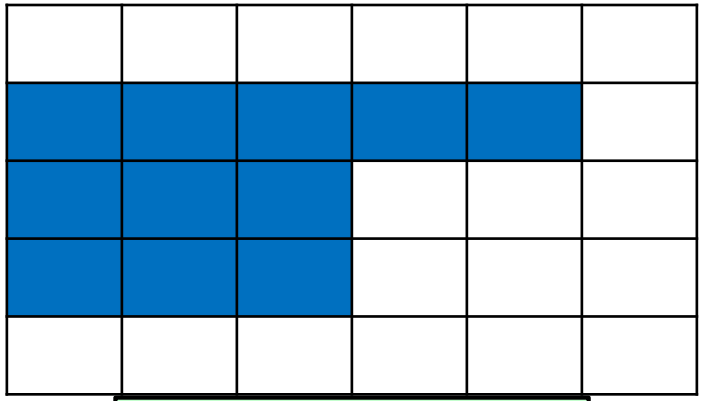
Manhattan Distance



example:
min(10, 7+2,8+1,8+2,**7+1**)=**8**

Calculation of Distance Map cont'd

- Example for Manhattan Distance. First run from top-left to bottom-right.



binary input image

∞	∞	∞	∞	∞	∞
0	0	0	0	0	∞
∞	∞	0	∞	∞	∞
0	0	0	∞	∞	∞
∞	∞	∞	∞	∞	∞

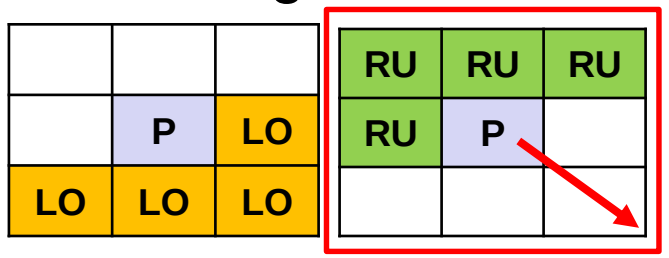
edges assigned weight 0

∞	∞	∞	∞	∞	∞
0	0	0	0	0	∞
∞	∞	0	∞	∞	∞
0	0	0	∞	∞	∞
∞	∞	∞	∞	∞	∞

result after first row

∞	∞	∞	∞	∞	∞
0	0	0	0	0	∞
1	1	0	1	1	2
0	0	0	1	2	3
1	1	1	2	3	4

final result for LO➔RU



Calculation of Distance Map cont'd

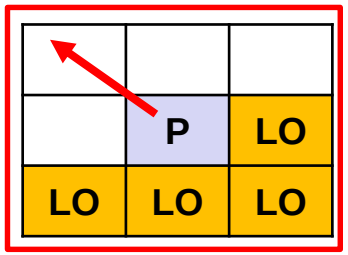
- Example for Manhattan Distance. Second run from bottom-right to top-left to complete the distance map.

∞	∞	∞	∞	∞	∞
0	0	0	0	0	1
1	1	0	1	1	2
0	0	0	1	2	3
1	1	1	2	3	4

processing lowest row first

∞	∞	∞	∞	∞	∞
0	0	0	0	0	1
1	1	0	1	1	2
0	0	0	1	2	3
1	1	1	2	3	4

result after second iteration



RU	RU	RU
RU	P	

1	1	1	1	1	2
0	0	0	0	0	1
1	1	0	1	1	2
0	0	0	1	2	3
1	1	1	2	3	4

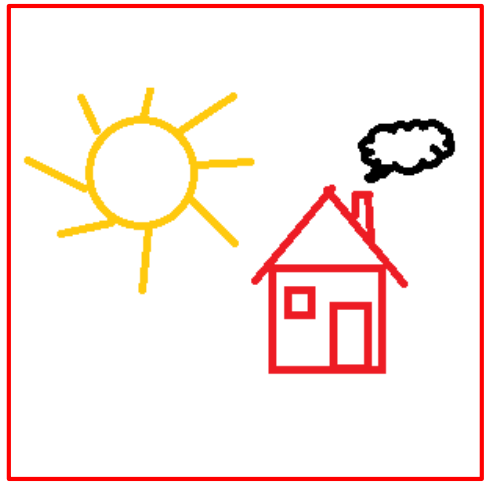
final result for bottom-right to top-left run

Mutual Information

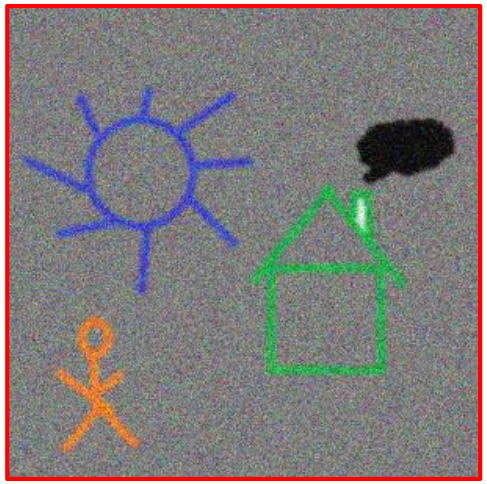
- if intensity profile, edges and morphology in general are not matching between images ***A*** and ***B***, calculation of Mutual Information (MI) still might be feasible.
  - thereby, the joint 2D histogram is calculated as basis for evaluating the level of correlation
  - mutual information defined as $I(A,B) = H(B) - H(B|A)$ with
    - H(A)... Shannon entropy calculated from the grey values in **A** with $H(A) = -\sum_i p_i \ln(p_i)$
    - H(A|B)... entropy according to conditional probabilities. I.e. Probability of observing b_i at position $B(x,y)$ already given a value a_i at position $\mathrm{A}(x,y)$.
    - instead, $H(A,B)$ of joint probabilities easier to handle, thus MI defined as $MI(A,B) = H(B) - H(B|A) = H(B) + H(A) - H(A,B)$
  - 2D histogram for graphic representation of MI.
    - compactness of the clusters indicate the level of correlation
    - in case of perfect match and congruent intensity profiles, the 2D histogram is manifested in the diagonal only showing the same characteristic as a 1D histogram of **A** or **B**.

Mutual Information cont'd

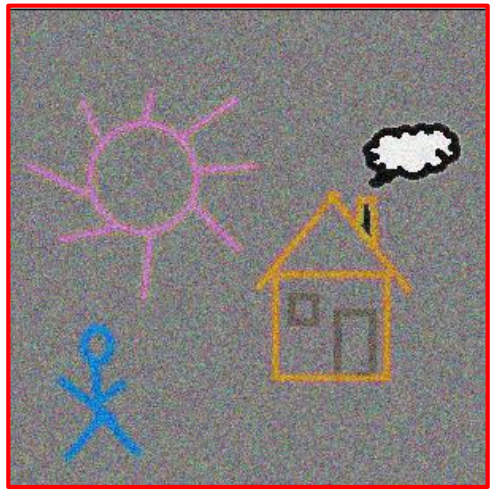
- example
  - neither intensity profile nor image morphology must match in a perfect way to evaluate image congruency utilizing MI metrics



A



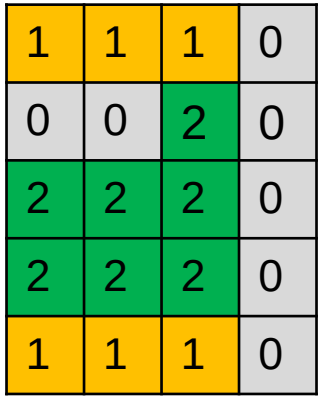
B



diff(A,B) – even if perfectly aligned, the difference image visualizes significant mismatches ➔ MI required

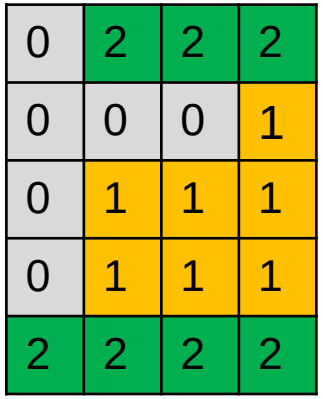
Mutual Information cont'd

- example



1	1	1	0
0	0	2	0
2	2	2	0
2	2	2	0
1	1	1	0

A



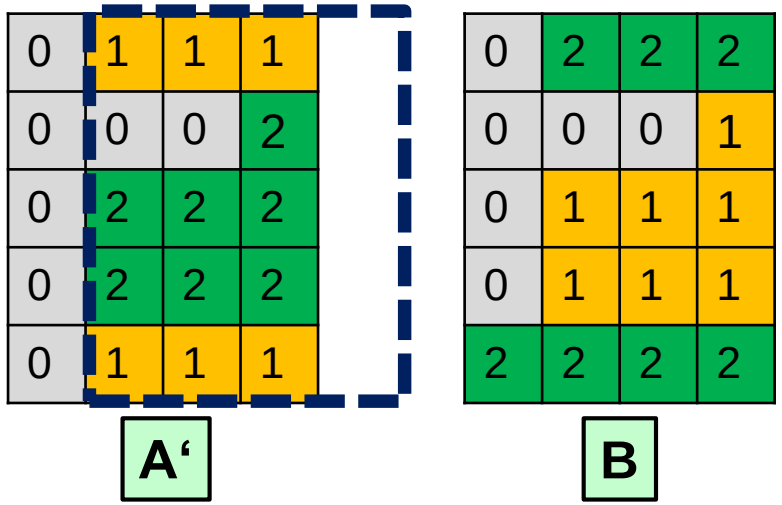
0	2	2	2
0	0	0	1
0	1	1	1
0	1	1	1
2	2	2	2

B

- entropy **A**:
  - p(0)=0.35
  - p(1)=0.3
  - p(2)=0.35
  - H(A')=1.096
- entropy **B**:
  - p(0)=0.3
  - p(1)=0.35
  - p(2)=0.35
  - H(B)=1.096
- entropy **A,B**:
  - p(0,0)=0.1
  - p(0,1)=0.15
  - p(0,2)=0.1
  - p(1,0)=0.05
  - p(1,1)=0
  - p(1,2)=0.25
  - p(2,0)=0.15
  - p(2,1)=0.2
  - p(2,2)=0
  - H(A,B)=1.847901
  - I(A,B)=**0.344234**

Mutual Information cont'd

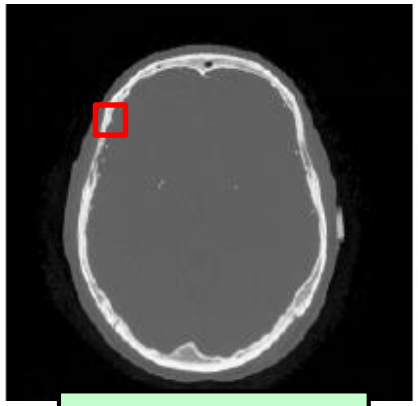
- example cont'd after reg:



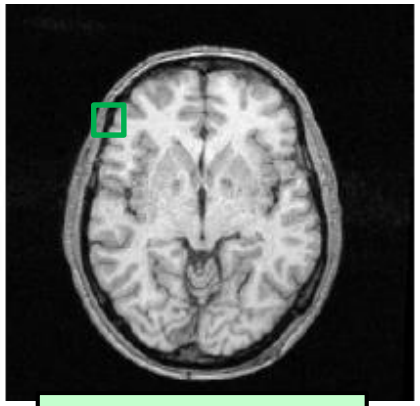
- entropy **A**:
  - p(0)=0.35
  - p(1)=0.3
  - p(2)=0.35
  - H(A')=1.096
- entropy **B**:
  - p(0)=0.3
  - p(1)=0.35
  - p(2)=0.35
  - H(B)=1.096
- entropy **A, B**:
  - p(0,0)=0.3
  - p(0,1)=0.0
  - p(0,2)=0.05
  - p(1,0)=0
  - p(1,1)=0
  - p(1,2)=0.3
  - p(2,0)=0
  - p(2,1)=0.35
  - p(2,2)=0
  - H(A,B)=1.436395
  - I(A,B)=**0.75574**

Mutual Information cont'd

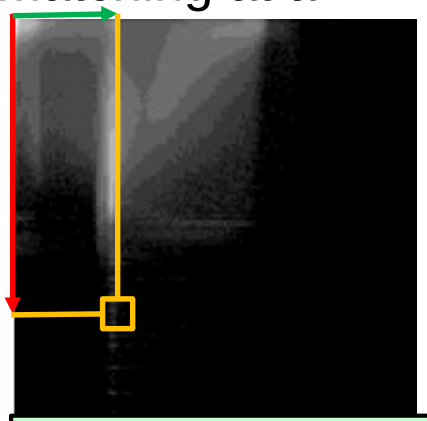
- MI highly relevant for multi-modal image analysis, e.g. in medicine
  - registration of MR and CT – intensity profile not matching at all



CT scan, **A**

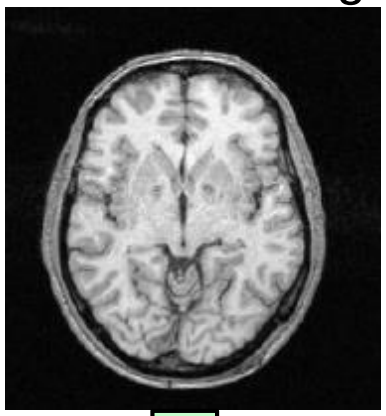


MRI scan, **B**

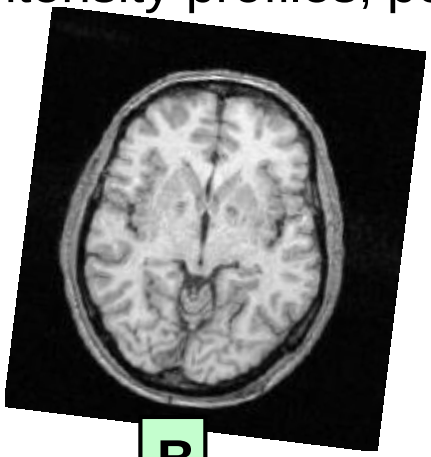


2D histogram, registered

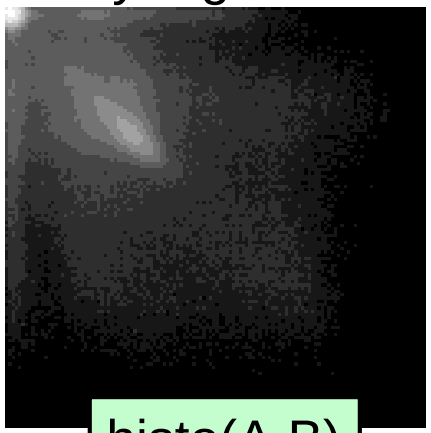
  - in case of matching intensity profiles, perfectly registered images show diagonal 2D histogram



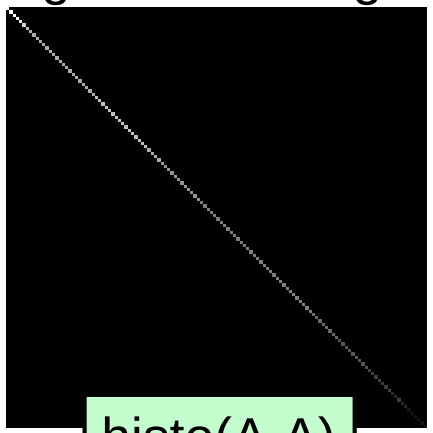
A



B



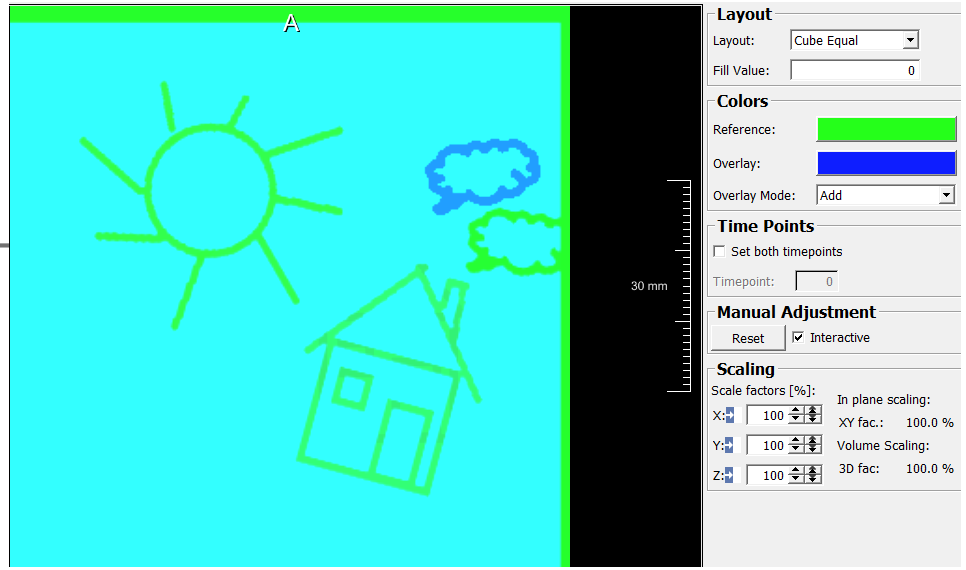
histo(A,B)

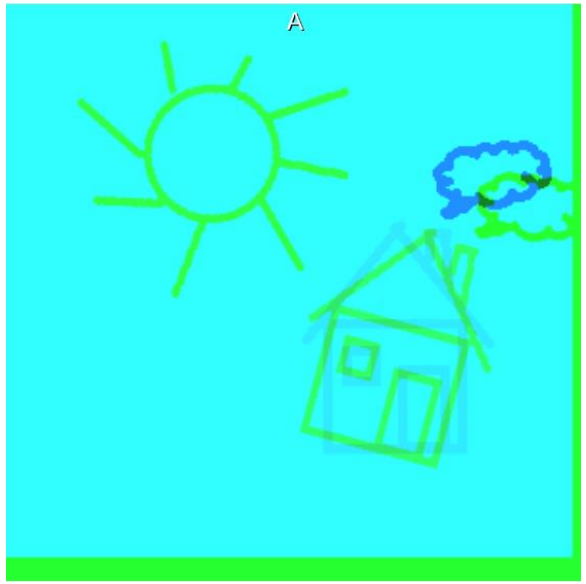


histo(A,A)

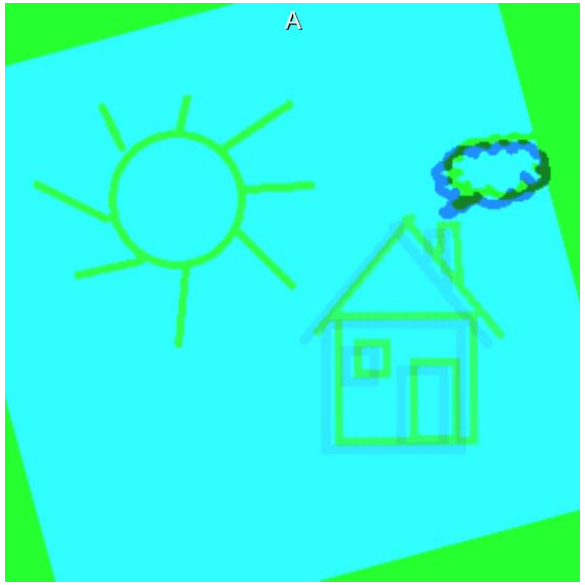
Manual Inspection

- utilizing LUT overlay, the level of congruence is evaluated utilizing visual inspection

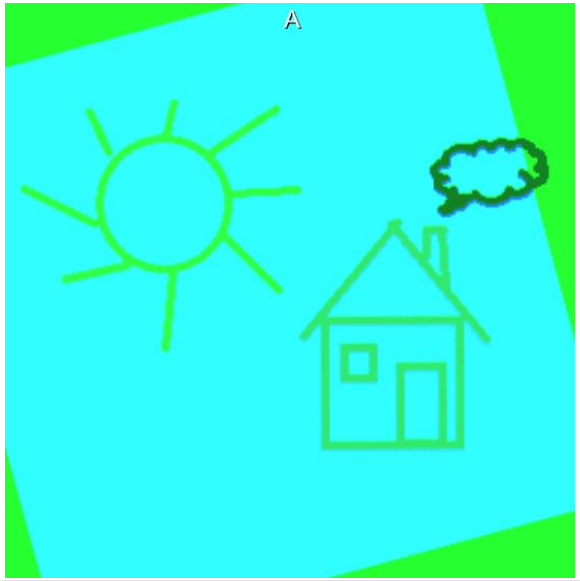




overlay **A, B** unregistered



overlay **A, B** registered w.r.t. rotation



overlay **A, B** registered w.r.t. rotation and translation