Life Sciences und
Facility Management

ICLS Institut für
Computational Life Sciences

ADLS, Image and Signal Processing, Midterm Exam II

Date: 29. April 2024
Lecturer: Norman Juchler

- Enter your name legibly below.
- Permitted aids: Writing utensils, two A4 pages of summary, calculator, dictionary
- Duration of exam **30 min**
- All sketches, calculations, derivations and considerations must be written on these sheets (front and back) and handed in. Additional sheets are not allowed.
- Provide answers in English
- Clearly cross out invalid answeres and results. If it is unclear which result applies, no points are awarded.
- Do not write with pencil, colored pencil or other erasable pens

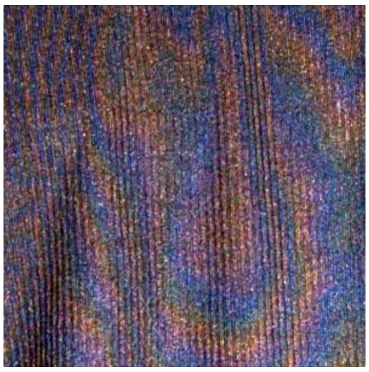
Family name: **First name:**

Exercise	1	2	3	4	Total
Maximum	10	9	10	7	36
Result					

Grade: ____________ **Good luck!**

Question 1

10 P.



a) **Color distortion** (3 P. possible)

The picture above shows a close-up of a fabric with an unfavorable camera setting. Explain the nature of this distortion. What is this effect called?

b) **Image quality attributes** (3 P. possible)

List three different aspects that influence the quality of a picture. Provide a name for each aspect or concept and describe it in one to two sentences.

c) **Color spaces** (3 P. possible)

RGB is the most common color space in digital image processing. Nevertheless, many other color spaces exist.

- Name three alternative color spaces. If you use acronyms, state what each letter means.
- Is RGB not good enough? Why are there other color spaces? Name at least two reasons.

d) **Memory footprint** (1 P. possible)

Suppose we are working with an RGBA image with four channels, 16-bit image depth and a resolution of 1920×1080px (full HD). How much memory (in bytes) is required for this image? Recall that 1 byte = 8 bits.

Solution 1

a) **Color distortion**

See lecture slides for SW08. The effect shown in the illustration is a *color moiré pattern*. It is caused by undersampling of the color channels. The photographed object contains patterns that are finer than the resolution of the camera. Remember the Nyquist-Shannon sampling theorem! Undersampling can lead to visible aliasing artifacts. In digital color images, the different color channels are not sampled at the same rate, such as with a Bayer filter. Aliasing can lead to different patterns in the different channels. Hence the colorful patterns.

b) **Image quality attributes**

See lecture slides for SW08.

c) **Color spaces**

- **HSV (Hue, Saturation, Value)**: Describes colors in terms of their hue (Farbton), saturation, and brightness. It is considered more intuitive than additive color mixing as with RGB and is often used in image editing software.
- **CMYK (Cyan, Magenta, Yellow, Key / black)**: Used in color printing. Represents colors as a combination of cyan, magenta, yellow, and black ink. It's a subtractive color model.
- **L*a*b* (CIELAB)**: A device-independent color space designed to approximate human vision. Describes colors with a lightness/luminance component (L*) and two color-opponent dimensions a* (green to red), and b* (blue to yellow). Is approximately perceptually uniform.
- **L*u*v* (CIELUV)**: Another perceptually uniform color space that is often used in color difference calculations.
- **XYZ (CIEXYZ)**: A standard based on human color perception. Represents colors using tristimulus values X, Y, and Z, which correspond to the stimulation of the three types of cones in the human eye.
- **YCbCr**: Used in video compression and broadcasting, separates image luminance (Y) from chrominance components (Cb and Cr).

c) **Color spaces** (cont.)

Although RGB is the most widely used color space, it is not always the best choice for image representation. There are several characteristics of color spaces that can play a role:

- Perceptual uniformity means that a change of length δ in any direction of the color space is perceived by a human as the same change. $\leftrightarrow$ CIELAB
- Device independence: RGB values can vary significantly between different devices (e.g., monitors, printers, cameras). $\leftrightarrow$ CIELAB, CIELUV, CIEXYZ
- Separation of color (chrominance) and intensity (luminance) information: The human visual system is more sensitive to changes in intensity than to changes in color. By separating color and intensity information, we can apply different processing steps to each component. $\leftrightarrow$ YCbCr, YUV

For some image processing tasks, it can be advantageous to change the color space. Histogram alignment, for example, leads to much more natural results if the luminance and chrominance information is separated.

d) **Memory footprint**

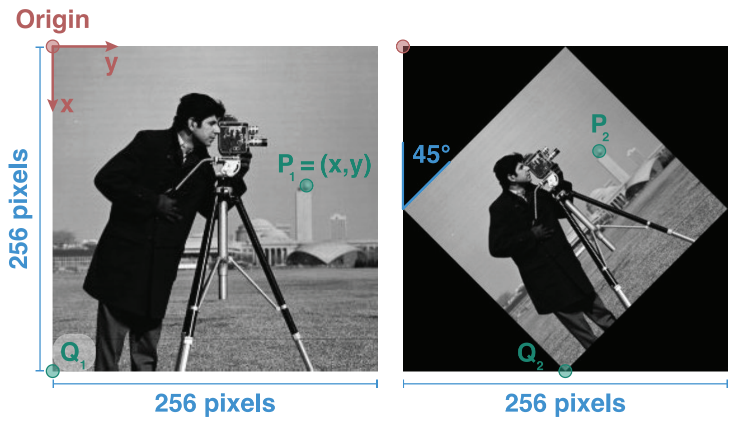
With $n = 4$ color channels, image depth $d = 16$bit, width $w = 1920$px and height $h = 1080$px, and under the assumption that no compression scheme is in place, the memory footprint M for the image is:

$$\begin{aligned} M &= n \cdot d \cdot w \cdot h \\ &= 132\ 710\ 400 \text{ Bits} \\ &= 16\ 588\ 800 \text{ Bytes} \\ &= 15.8 \text{ MB} \end{aligned}$$

Recall that 1 Byte = 1B = 8 Bit, and 1kB = 1024 Bytes

Question 2

9 P.



a) **Affine transformations** (7 P. possible)

Take a look at the image above. The image is 256x256 pixels in size and the angle of rotation is $+45°$. Answer the following questions:

1. What transformations are required to convert the left image into the right image. Suggest one possible sequence of elementary transformations to achieve this. Specify the transformations in words.
2. How many such sequences are possible?
3. Write the affine transformation matrix $A \in \mathbb{R}^{3\times3}$ as a product of elementary transformations. Specify the parameters of the elementary transformation matrices correctly, but you do not have to calculate A explicitly.
   Note: The origin of the coordinate system is in the top left corner!
4. The point $Q_1 = (256, 0)$ will be mapped to $Q_2 = (256, 128)$. Can you confirm this with your formula for A? How would you proceed if you did not find a formula for A?

b) **Homogeneous coordinates** (2 P. possible)

If we have a point $P = (u, v)$, what is its representation using homogeneous coordinates? What is the reason for using homogeneous coordinates?

Solution 2

a) **Affine transformations**

1. One (minimal) sequence of elementary transformations is:
   - Translate image such that center falls onto origin
   - Rotate about $45°$ counterclockwise, around origin
   - Scale by a factor of $s = \frac{1}{\sqrt{2}}$ with respect to origin $^{\triangle}$
   - Translate image back such that center of image again is at point $p = (\frac{w}{2}, \frac{w}{2})$

   Alternatively, one can chain the following transformations:
   - Rotate by $45°$ around center of the image $p = (\frac{w}{2}, \frac{w}{2})$
   - Scale the image $\frac{1}{\sqrt{2}}$ with respect to center of the image $^{\triangle}$

   However, the rotation around a point that is not the origin cannot be represented by an elementary transformation matrix (see primer on affine transformations).

   $^{\triangle}$ To understand the choice of the scale factor, see that the image must be scaled such that the diagonal of length $d_1 = w\sqrt{2}$ in the original image has length $d_2 = w$ after scaling. Hence the factor must be $s = \frac{1}{\sqrt{2}}$.

2. There are infinitely many transformation sequences possible that yield the same result. For instance, one can divide a translation into an arbitrary number of sub-steps.

3. The minimal sequence of transformations from 1) can be expressed as follows:

   $$A = T_{C \leftarrow O} \cdot S \cdot R \cdot T_{O \leftarrow C} \tag{1}$$

   This is a product of elementary transformation matrices. Note that the first transformation (translation from image center to origin) appears on the very right. The different transformation matrices are defined as follows ($w = 256$):

   $$T_{O\leftarrow C} = \begin{pmatrix} 1 & 0 & -w/2 \\ 0 & 1 & -w/2 \\ 0 & 0 & 1 \end{pmatrix} \qquad R = \begin{pmatrix} \cos 45° & -\sin 45° & 0 \\ \sin 45° & \cos 45° & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

   $$T_{C\leftarrow O} = \begin{pmatrix} 1 & 0 & +w/2 \\ 0 & 1 & +w/2 \\ 0 & 0 & 1 \end{pmatrix} \qquad S = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

   We use homogenized coordinates here. The resulting transformation matrix A can be used to see where a pixel at position $x_H = (x_1, x_2, 1)$ in the original image I is found in the transformed image J:

   $$y_H = A \cdot x_H$$

4. The task is to verify that at $Q_1 = (256, 0)$ will be mapped to $Q_2 = (256, 128)$. Using the homogeneous representation of these coordinates, $x_H = (256, 0, 1)^T$, and $y_H = (256, 128, 1)^T$ we need to show that $y_H = A \cdot x_H$ indeed holds true.

   There are at least two possible approaches:

   i Compute the matrix A according to equation (1) and then compute $A \cdot x_H$

   ii Apply one transformation after the other to x_H:

$$y_H = \underbrace{T_{C\leftarrow O} \cdot \underbrace{\left(S \cdot \underbrace{\left(R \cdot \underbrace{(T_{O\leftarrow C} \cdot x_H)}_{y_1}\right)}_{y_2}\right)}_{y_3}}_{y_H = y_4}$$

   The approach i requires quite a few calculations and is prone to errors if performed manually. We therefore follow ii:

   Step 1: Translation

$$y_1 = T_{O\leftarrow C} \cdot \begin{pmatrix}256\\0\\1\end{pmatrix} = \begin{pmatrix}1 & 0 & -128\\0 & 1 & -128\\0 & 0 & 1\end{pmatrix} \cdot \begin{pmatrix}256\\0\\1\end{pmatrix} = \begin{pmatrix}256-128\\-128\\1\end{pmatrix} = \begin{pmatrix}128\\-128\\1\end{pmatrix}$$

   Step 2: Rotation

$$y_2 = R \cdot y_1 = \begin{pmatrix}\cos 45^\circ & -\sin 45^\circ & 0\\ \sin 45^\circ & \cos 45^\circ & 0\\ 0 & 0 & 1\end{pmatrix} \cdot \begin{pmatrix}128\\-128\\1\end{pmatrix}$$

$$= \begin{pmatrix}\frac{2}{\sqrt{2}} & -\frac{2}{\sqrt{2}} & 0\\ \frac{2}{\sqrt{2}} & \frac{2}{\sqrt{2}} & 0\\ 0 & 0 & 1\end{pmatrix} \cdot \begin{pmatrix}128\\-128\\1\end{pmatrix} = \begin{pmatrix}128\sqrt{2}\\0\\1\end{pmatrix}$$

   Step 3: Scaling

$$y_3 = S \cdot y_2 = \begin{pmatrix}\frac{1}{\sqrt{2}} & 0 & 0\\ 0 & \frac{1}{\sqrt{2}} & 0\\ 0 & 0 & 1\end{pmatrix} \cdot \begin{pmatrix}128\sqrt{2}\\0\\1\end{pmatrix} = \begin{pmatrix}128\\0\\1\end{pmatrix}$$

   Step 4: Translation

$$y_H = T_{C\leftarrow O} \cdot y_3 = \begin{pmatrix}1 & 0 & 128\\0 & 1 & 128\\0 & 0 & 1\end{pmatrix} \cdot \begin{pmatrix}128\\0\\1\end{pmatrix} = \begin{pmatrix}128+128\\128\\1\end{pmatrix} = \begin{pmatrix}256\\128\\1\end{pmatrix}$$

   We have thus shown that $x = (256, 0)$ is actually mapped to $y = (256, 128)$.

b) **Homogeneous coordinates**

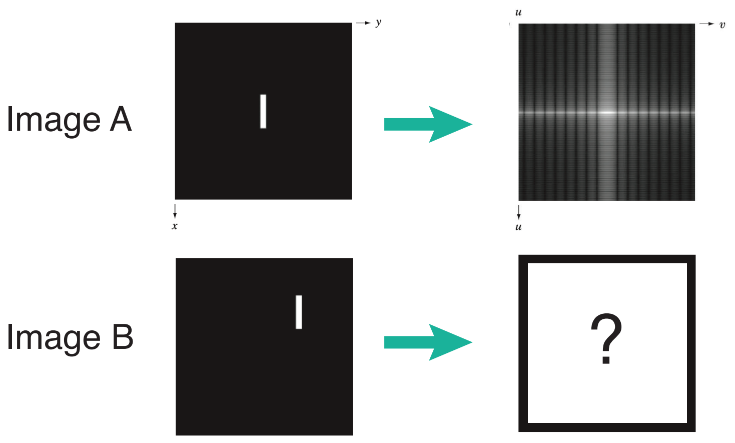
The homogeneous representation of P is as follows:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix}$$

The main reason for using homogeneous coordinates is to represent translations as matrix multiplications and to represent arbitrary affine transformations as matrix multiplications. In the expression $y_H = A \cdot x_H$, the matrix A contains all transformation parameters, x_H represents the coordinates of the input pixel, and y_H the coordinates of the transformed pixel.

Question 3

10 P.



a) **Translation and amplitude spectrum** (2 P. possible)

In the first row of the above figure, you can see the (logarithmic) amplitude spectrum for input image A. What happens to the amplitude spectrum if the input image is shifted by $t = (\Delta x, \Delta y)$ as seen in image B? Explain!

b) **Fourier spectrum**, characteristics (4 P. possible)

Assume that the input image has width w and height h. Answer the following questions about the DFT and the amplitude spectrum of that image.

- What are width and height of the resulting image that represents the amplitude spectrum?
- Where can we typically read the DC component?
- Are there any symmetries for real-valued input images? If yes, which ones?
- What is the largest spatial frequency that we can read from the spectrum?

c) **Fourier spectrum**, implementation (4 P. possible)

Write a Python function `compute_spectrum(img)` that takes an image as input, calculates its amplitude spectrum and displays it. The code does not have to be functional, but the relevant steps should be recognizable. Comment on what happens in each step.

Solution 3

a) **Translation and amplitude spectrum**

The amplitude spectrum of the translated image will be the same.

Explanation: The amplitude is shift-invariant! Recall that for the translated version of the image $f(x, y)$, only the phase will be affected by the translation, whereas the amplitude of the Fourier transform $F(u, v)$ will be the same. It holds the following:

$$f(x - x_0, y - y_0) \leftrightarrow F(u, v) \cdot e^{-2\pi i\left(\frac{ux_0}{M} + \frac{vy_0}{N}\right)} = z_1(u, v) \cdot z_2(u, v)$$

In detail: The expression on the right side represents a multiplication of two complex numbers. First, recall that every complex number $z \in \mathbb{C}$ can be written in polar form with magnitude $r = |z|$ and phase $\varphi = \arg(z)$:

$$z = r \cdot e^{i\varphi}$$

For the product of two complex numbers, it holds that

$$z_1 \cdot z_2 = |z_1| \cdot |z_2| e^{i(\varphi_1 + \varphi_2)}$$

In words: the amplitudes are multiplied and the phases are added. If you cannot follow, please read the primer on complex numbers!

Now note that the amplitude of $z_2 = e^{-2\pi i\left(\frac{ux_0}{M} + \frac{vy_0}{N}\right)}$ is 1. For the amplitude of the shifted image, the following therefore applies

$$|F(u, v)| \cdot \left| e^{-2\pi i\left(\frac{ux_0}{M} + \frac{vy_0}{N}\right)} \right| = |F(u, v)|$$

The Fourier amplitude of the shifted image equals the Fourier amplitude of the original image.

Note that the amplitude spectrum will still change if the edge is affected by the shift or if the image content is cropped.

b) **Fourier spectrum**, characteristics

- The width and hight of the amplitude image also have width w and h.
- The DC component can be read in the center of the amplitude image at $\left(\frac{h}{2}, \frac{w}{2}\right)$.
- With real-valued input data, the amplitude spectrum exhibits point symmetry with respect to the central point $\left(\frac{h}{2}, \frac{w}{2}\right)$.
  Detail: For a real-valued image $f(x, y)$ with Fourier transform $F(u, v)$, it holds:
  $$\begin{aligned} F(-u,-v) &= \overline{F(u,v)} \qquad \text{(conjugate symmetry of the FT)} \\ |F(-u,-v)| &= \left|\overline{F(u,v)}\right| = |F(u,v)| \end{aligned}$$
  Hence, the amplitude spectrum is point symmetric w.r.t. to the central point.
- The largest spatial frequency represented in an amplitude image is $\max(\frac{h}{2}, \frac{w}{2})$, or the larger of the two Nyquist frequencies for the two axes.

c) **Fourier spectrum**, implementation Find below the code required to calculate and display the amplitude spectrum of an image.

```python
import numpy as np
import scipy.fft as fft
import matplotlib.pyplot as plt

def compute_spectrum(img):
    F = fft.fft2(img)          # Compute the DFT, results in an array of complex values
    F = fft.fftshift(F)        # Reorder the frequencies from negative to positive
    F_abs = np.abs(F)          # Compute the magnitude: sqrt(Re(F)**2 + Im(F)**2)
    F_abs = np.log(1+F_abs)    # To better recognize the patterns in the amplitude image
                               # Add 1 to avoid log(0)
    F_abs /= F_abs.max()       # (Optional step, normalize image)
    plt.imshow(F_abs, cmap='gray')
```

Question 4

7 P.

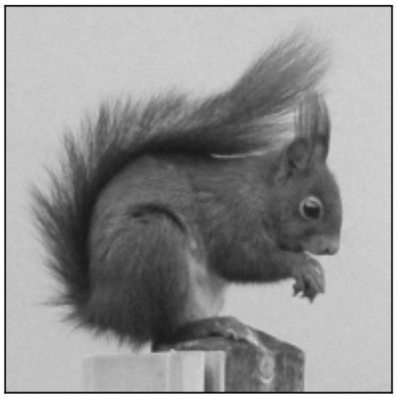


Image A

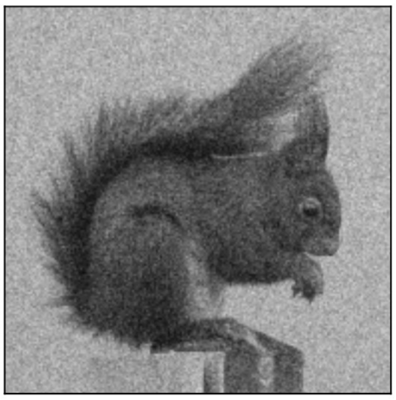


Image B

a) **Edge detection** (5 P. possible)

Assume you want to identify the edges of the above image A using two Sobel filters (in x- and y-direction).

- Which are the steps to get the results?
- For every step, write down the corresponding Python function / expression (use numpy and scipy only).
- Are any kernels involved? How would they look like?

b) **Edge detection and noise** (2 P. possible)

Now, assume that you want to extract the edges for image B. Which are your best options in this case?

Solution 4

a) **Edge detection**

We can use the gradient magnitude to detect edges in an image. Sobel filtersăare commonly used to estimate the gradient of an image. Once we know the gradient magnitude for each pixel, we can apply a threshold to detect edges. All this involves the following steps:

- Approximate the partial derivatives of the image I in horizontal and vertical direction using Sobel filters.
- Practically, this is done using convolution. If S_x and S_y are 3×3 Sobel kernels to estimate the partial derivative in x- and y-direction, then:

$$\frac{\partial I}{\partial x} \approx S_x * I$$
$$\frac{\partial I}{\partial y} \approx S_y * I$$

  Here, the following kernels are used:

$$S_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \qquad S_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

- The gradient is defined as $\nabla I(x,y) = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)^T$.
- Compute the magnitude $M = |\nabla I(x,y)|$ and compare it to a threshold $M > \phi$.
- The corresponding code

```
import numpy as np
import scipy.signal as signal


dIx = signal.convolve2(img, Sx, ...)
dIy = signal.convolve2(img, Sy, ...)
grad = np.sqrt(dIx**2 + dIy**2)
edges = grad > mag
```

b) **Edge detection and noise**

Noise will be amplified by the gradient operator. To reduce the noise, we can apply a low-pass filter before computing the gradient:

$$K_G = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$
$$I_{smooth} = K_G * I$$

We then proceed using I_{smooth} to compute the gradients.

Alternatively, we can also compute the derivative of the Gaussian function, approximate it with a $k \times k$ kernel and compute the derivative of the smoothed image in one go. See lecture slides on *Features and Segmentation (SW10)*.