Mathematisch-Naturwissenschaftliche
Fakultät

Fachbereich Informatik
Arbeitsbereich Visual Computing

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Medical Visualization

Prof. Dr.-Ing. Thomas Küstner

WS 2025/2026

Assignment 3

Remarks

The exercises should be completed in groups of three to four students. Please submit your exercises in ILIAS *before* 23:55 on the closing date. At *least* one member of the group *must* be present at the exercise group meeting, being prepared to explain *each* exercise. Random groups will be asked to present their solutions.

Python

- **Please put the answers to comprehension questions into a separate PDF file.**
- **If there is a built-in function for an algorithm you are supposed to implement, do *not* use it.**

VTK

For the programming tasks, Python is used together with the Visualization Toolkit (VTK).
You will find the necessary information about the required classes and functions here:

- Python Examples: **https://examples.vtk.org/site/Python/**
- Reference to all functions: **https://vtk.org/doc/nightly/html/index.html**
  (If you view a class description, be sure to click one the 'More...' right at the top)
- Documentation: **https://docs.vtk.org/en/latest/**

LLMs

We strictly do not allow the use of AI to solve entire exercises. Solutions or code submissions that have been entirely generated by LLMs like ChatGPT or Claude will be graded with 0 points.

Submission

Create a folder **lastname1_..._lastnameN** with your team members last names in alphabetical order. Copy your files into this folder. Hand in all of your solutions and files necessary for the code to run (no need to upload the exercise sheet). Create a zip file **lastname1_..._lastnameN.zip** from the folder. Submit the zip-file to Ilias.

Exercise 1: Lecture Questions (10 points)

Answer the following questions briefly and precisely:

a) We want to capture an MRI of a patients brain and especially make the fluids visible in the resulting scan. Do we need to capture a T1 weighted or T2 weighted image? Explain your answer and also describe what repetition time (TR) and echo time (TE) are and how different timings result in differently weighted images.

b) Create a sketch of the complex number $z := \exp(j\alpha) \in \mathbb{C}$. List both the real and the imaginary part of z

c) How does the Fourier transform of a linear combination of two functions f and g look like (i.e. $a \cdot f + b \cdot g$)?

d) What is the convolution theorem of the Fourier transform and how can it be applied when filtering signals?

e) We are sampling a signal with a sampling frequency of f_{sample}. If we want to avoid aliasing in the reconstruction of the signal, how do we need to set the frequency f_{limit} of the band-limit we apply to the signal before sampling?

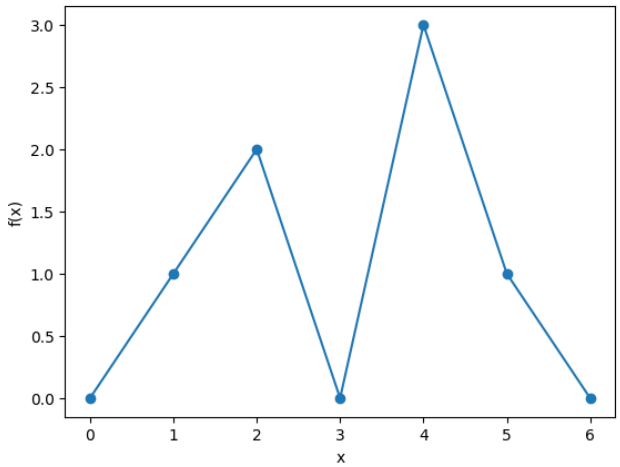
f) What is the difference between the discrete-time Fourier transform (DTFT) and the discrete Fourier transform (DFT)? How do they relate to each other?

g) In what situations would a median filter improve the image quality? Is this a linear or nonlinear filter? Is this filter edge-preserving or not? Briefly explain your answers.

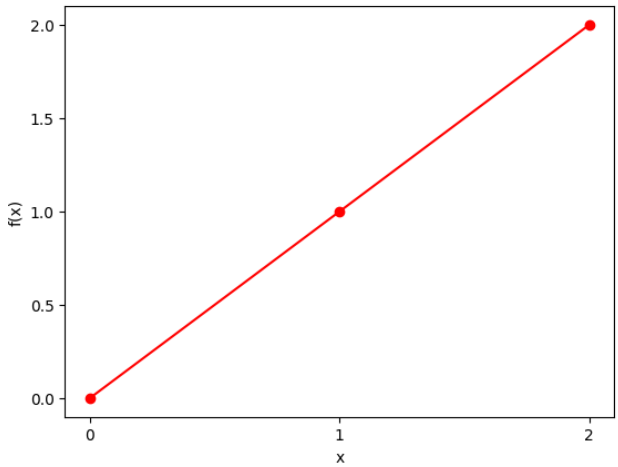
h) What is the difference between uncertainty and confidence? Explain both concepts and describe what aleatoric uncertainty and epistemic uncertainty are.

i) In machine learning, what does is mean to train in a supervised or unsupervised setting? Explain and name an example task for each setting.

j) Below in Fig 1a you can see a one-dimensional signal that has been sampled with $n = 7$ samples. Sketch the plot of the result of convolving this signal with the kernel given in Fig 1b. You may ignore the border cases and only compute in the regions fully covered by the kernel.



(a) Discrete Signal



(b) Kernel

Figure 1: Discrete signal and kernel for task j)

General Information

The goal of this assignment is to become familiar with the Fourier transform and its applications. You will implement the discrete Fourier transform in Python using numpy and apply it to an example. If you are unable to implement the transform successfully, you may use the numpy functions **np.fft.fft** and **np.fft.ifft** for the tasks where you need them. In addition to these, you will also need the functions **np.fft.fftshift**, **np.fft.fft2**, and **np.fft.ifft2**. The function **np.fft.fftshift** is used so that the frequencies are plotted at the expected positions (i.e. frequency 0 is at the center of the plot) and **np.fft.fft2** implements the 2D Fourier transform.

Please complete the following tasks in the files **"exercise_02.py"** and **"exercise_03.py"** respectively.

Exercise 2: Discrete one-dimensional Fourier Transform (10 points)

The formula for the discrete Fourier transform (DFT) $F = \mathcal{F}(f)$ for a sampled function f is defined as

$$F(u) = \sum_{x=0}^{N-1} f(x) \cdot \exp\left(-\frac{2\pi j}{N} \cdot ux\right)$$

and for the inverse transform $f = \mathcal{F}^{-1}(F)$

$$f(x) = \frac{1}{N} \sum_{u=0}^{N-1} F(u) \cdot \exp\left(\frac{2\pi j}{N} \cdot ux\right)$$

a) Complete the function **fourier_transform** that performs the discrete Fourier transform or the inverse transform for a one-dimensional vector. The function receives two arguments: one is an input **f** containing a sampled signal or its Fourier transform and the other is a flag **inv** that determines whether the Fourier transform or the inverse Fourier transform should be performed.
It is possible to completely avoid loops in the implementation by vectorizing the equations. It makes the entire computation much faster but it is not a requirement to obtain full points.

b) The implemented function should now be applied to a function **f** composed of two overlapping cosine waves of different frequencies. This sampled function **f** is already provided in the script. Afterward, the function **f** and the Fourier transform **F** should both be plotted. To ensure that the frequencies are plotted at the expected location in the Fourier plot, the function **np.fft.fftshift** can be applied to the Fourier transform **F**.

c) Next we will work with a modified version of **f** in which noise has been added to the signal. This noisy signal **f_noisy** is again provided in the script. The noisy signal **f_noisy** should then be smoothed using a moving average filter to reduce the noise. Implement a method called **box_filter** and use it to smooth the signal.
The filter should be applied using convolution (implement it yourself, don't use available functions). Plot both the filtered signal and its Fourier transform (to avoid issues at the edges of the signal, you can pad it with a zero on both sides using **np.pad**). As the filter kernel, you can use the provided kernel $\frac{1}{4} \cdot (1 \quad 2 \quad 1)$. How would you rate the filtered result? (explain either as comment or in a separate PDF)

d) After attempting to smooth the signal using a box filter, the noise should be removed as effectively as possible using the Fourier transform **F_noisy** of the noisy signal **f_noisy** from the previous task. The original signal should then be reconstructed using the inverse Fourier transform.
Consider how you could remove the noise in the Fourier domain and implement your method in the function **ft_denoise**. Plot both the reconstructed signal **f** and its Fourier transform **F** and explain your method (either as comment or separate PDF).

e) Which method was better at reducing the noise? Explain both your answer and the reason why the method performed better. Also explain when this method might not be applicable.

Exercise 3: Discrete 2D Fourier Transform (10 points)

For the second part, we will look at a two-dimensional signal **f_quad** and the 2D discrete Fourier transform.

a) Plot both the function/image **f_quad** and its Fourier transform **F_quad**. You may use **np.fft.fft2** to compute the Fourier transform.

b) Similar to the one-dimensional example, implement a function **box_filter_2d** that applies a filter kernel using 2D convolution (implement it yourself!). To avoid issues at the border, you may use **np.pad** again to add zeros to all sides.
Now filter the signal **f_quad** once using a low-pass filter and once using a high-pass filter. You may use the provided gaussian filter kernel for the low-pass filter

$$G = \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

and the provided kernel inspired by the Laplace operator for the high-pass filter

$$L = \begin{pmatrix} -1 & -2 & -1 \\ -2 & 12 & -2 \\ -1 & -2 & -1 \end{pmatrix}$$

Plot both the low-passed and high-passed signals as well as both of their Fourier transforms.

c) Write two functions **low_pass_filter** and **high_pass_filter**, which implement ideal low-pass and high-pass filters in the frequency domain based on a frequency **threshold** (ideal means it has a hard frequency cutoff).
Apply these filters to the Fourier transform of the function **f_quad**. Plot the resulting filtered signal as well as their Fourier transform for each case and explain the results. Additionally, compare them with the results from the previous task.