1 Preparation

1.1 Software installations

Install the required software on a workstation of your choice. You need for the lecture:

1. A git client, e.g. turtoise git under Windows or git under Linux.
2. Python 3.11 or higher
3. A Latex installation, e.g. miktex and texworks under Windows or a corresponding distribution under Linux
4. An installation of Jupyter Lab

1.2 Create and Register Repository (5 points)

Check out the exercise file `inpainting_ss2024_exercise_01.zip` from your canvas course. Additional exercise files with exercise sheets and associated data and sample solutions are checked in during the semester. You can download the latest version after each lecture to gain access to the task sheet and associated code fragments or sample data.

Now create a publicly accessible repository for your solutions. Name it as follows:
`inpainting_ss2024_<lastname>_<firstname>`, e.g. `inpainting_ss2024_Dahmen_Tim`.
In this repository you push your solutions to the tasks. You can branch, push and merge as often as you want, the last commit of the main branch is evaluated.

Now copy the files from the exercise file to your solution repository. Add the files to version control.

Please edit your tasks by adding each:

1. Add the corresponding answers to the latex document with the questions. Please push both the Latex document and a compiled pdf to your solution repository. Please do not rename the task file. However, you can create additional files and link or otherwise use them to structure your solution.
2. Edit the programming tasks by adding to the existing source code files. Please do not rename them. Again, you can add your own files or subdirectories and include these new files to structure your solution.

Send an email to Tim.Dahmen@hochschule-aalen.de and let me know that you have created your repository.

I needed the following time to complete the task:

2 Data representation for sparse images

2.1 Advantages and disadvantages of different representations (5 points)

Describe in your own words the advantages and disadvantages of encoding sparse images a) using *nan* encoding b) using explicit positions (x,y,r,g,b) or c) using image and mask.

3 Interpolation

3.1 Linear interpolation (5 points)

Complete the file (linear_interpolation.ipynb). Some unit tests and some code are already included in the task file.

a) Add unit tests that test the case for boundary values.

b) Add code for linear interpolation of gray levels.

c) Add unit tests to ensure the correct handling of color values.

b) Add code for linear interpolation of color values (rgb).
Result grayscale:

Result RGB color:

I needed the following time to complete the task:

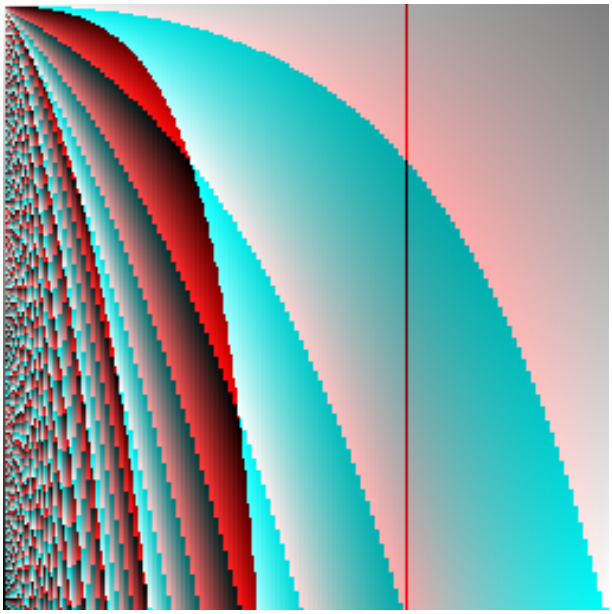
3.2 Bilinear interpolation (10 points)

Complete the file (linear_interpolation.py). Some unit tests and some code are already included in the task file.

a) Add unit tests to test the bilinear interpolation.

b) Describe your test strategy (i.e. which values do you test and why). Argue: does it make sense at this point to test cases outside the range of values (e.g. ¡0, ¿255)? Does it make sense to test for exceptions using unittest?

c) Add code for bilinear interpolation of grayscale. Result RGB color:



Run the program and recompile the Latex to display the results.

I needed the following time to complete the task:

3.3 Bonus Task (10 points)

Prove or disprove: In bilinear interpolation, it does not matter whether interpolation is first in the x and then in the y direction or vice versa.

I needed the following time to complete the task: