Project work: Title

Module: Image and Signal Processing (ISP-AD23-FS25)
Authors: Philly Filter and Conny Convolution
Date: DD.MM.YYYY

Introduction

Instructions: *Use this section to introduce and motivate your project. Provide enough context for fellow students to understand your goals.*

- *What problem are you addressing, and why is it relevant?*
- *What kind of image or signal data are you working with?*
- *What is the main goal of your processing task?*

Markdown: *If you're new to Markdown, the following resources might help:*

- *Quick overview*
- *GitHub-style Markdown*
- *Tutorial for Jupyter Notebooks*

Hint: *You can include simple HTML in Markdown cells (like the* `<span>` *blocks here) to improve formatting – but this is optional.*

Setup

Instructions: *This section is about configuring the Jupyter Notebook. You don't need to do much here – just make sure everything runs correctly.*

In [ ]:
```
# Basic imports
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

import cv2 as cv
import PIL
# ...

# Enable vectorized graphics
%config InlineBackend.figure_formats = ["svg"]

# Inline backend configuration
%matplotlib inline

# Setup plotting
PALETTE = [ (0.341, 0.648, 0.962, 1.0),
            (0.990, 0.476, 0.494, 1.0),
            (0.281, 0.749, 0.463, 1.0),
            (0.629, 0.802, 0.978, 1.0),
            (0.994, 0.705, 0.715, 1.0),
            (0.595, 0.858, 0.698, 1.0),
            (0.876, 0.934, 0.992, 1.0),
            (0.998, 0.901, 0.905, 1.0),
            (0.865, 0.952, 0.899, 1.0) ]

# For more color palettes, see here:
# https://seaborn.pydata.org/tutorial/color_palettes.html
# https://matplotlib.org/stable/users/explain/colors/colormaps.html
#PALETTE = sns.color_palette("husl", 8)
#PALETTE = sns.color_palette("viridis", 10)

print("Our color palette:")
sns.palplot(PALETTE, size=0.5)

sns.set_style("whitegrid")
plt.rcParams["axes.prop_cycle"] = plt.cycler(color=PALETTE)
plt.rcParams["figure.dpi"] = 300    # High-res figures (DPI)
plt.rcParams["pdf.fonttype"] = 42   # Editable text in PDF
```

Implementation

Instructions: *In this section, explain how you implemented your solution, focusing on the key processing steps and how they were translated into code. Justify your design choices and parameter settings, and describe any challenges you encountered during development. Structure this part like a tutorial to make it easy for fellow students to follow and learn from your approach.*

- *Which key functions, algorithms, or libraries (e.g., NumPy, OpenCV, librosa, etc.) did you use?*
- *How is your processing pipeline structured? (e.g., filtering → transformation → output)*
- *Explain any relevant design decisions (e.g., kernel size, interpolation method, threshold values).*
- *If you implemented parts yourself (e.g., a filter or transform), briefly explain how.*

Hint: *To keep your code clean and modular, encapsulate your processing or feature (as well as the intermediate processing steps) into functions. See the example below for illustration.*

In [ ]:
```
# Implementation
...

def processing_step_A(data):
    # Your processing logic goes here...
    ...
    return processed_data


def processing_step_B(data):
    # Your processing logic goes here...
    ...
    return processed_data


def apply_my_funny_feature(data, param1=10, param2=42):
    # Your processing logic goes here...
    ...
    data = processing_step_A(data)
    result = processing_step_B(data)
    ...
    return result
```

Results

Instructions: *Demonstrate the outcome of your solution. This section can be brief, but should show what your implementation produces.*

- *Use representative input and output examples to illustrate the effect of your processing.*
- *Don't limit yourself to best-case results – also show cases where the method performs poorly or struggles.*

In [ ]:
```
# Run your solution
data = ... # Load your data
result = apply_my_funny_feature(data, param1=10, param2=42)

# Visualize the result
...
```

Discussion

Instructions: *Reflect on your results and the overall performance of your solution. This section helps you and others understand what worked well, what did not, and why.*

- *Interpret your results: What patterns or weaknesses did you observe?*
- *How robust is your solution to variations in the input (e.g., noise, contrast, lighting)?*
- *Were there cases where your method failed or produced unexpected results? Why?*
- *What could be improved or extended in a future version?*
- *If applicable, compare your method to alternatives or standard approaches.*

Be honest and analytical – this section is not about perfection, but about insight.

Appendix

How to convert a Jupyter Notebook into a PDF

- *Run the entire Jupyter notebook and save it.*
- *Open a terminal and run the following command.*
  `jupyter nbconvert --to html "path/to/your/notebook.ipynb"`.
- *This creates a file notebook.html in the current working directory.*
- *Open the document in a web browser (the Opera browser works best, as it saves single-page PDFs!)*
- *Save as PDF*