Introduction

1 General Notes

This exercise is intended to practice the material discussed in the computer vision course. The lecture follows the book by Szeliski: "Computer Vision: Algorithms and Applications", which is freely available online [1]. It is highly advised that you read the book, especially the chapters which are covered in this course. Apart from this, we also recommend the book "Multiple View Geometry in Computer Vision" by Richard Hartley, a sample chapter of which is available online [2]. Also, the university library offers a pdf version of this book, so you can get access to it from there.

The exercises are structured as follows:

- There are 5 exercise sheets. They can give up to 10% of bonus points for the final exam. Also material from the curriculum of the exercises will end out in the final exam.
- Correct submissions guaranty 8% bonus, 2% bonus can be gained with qualitative aspects (complexity, style, etc.)
- The exercise sheets are going to be released on StudOn on Monday. The usual deadline is 2 weeks after release.
- You may work on the exercises individually or in groups of 2. You can use StudOn forum to find a group partner.
- The solutions should be uploaded to StudOn as single zip file before the specified deadline.
- Make sure to add your group partners when uploading your submission. inside the code.

[1] http://szeliski.org/Book/
[2] https://www.cambridge.org/core/books/multiple-view-geometry-in-computer-vision/0B6F289C78B2B23F596CAA76D3D43F7A

2 Compile and Run

The exercises are written in Python and use the open-source library OpenCV. You may work on your own computer, on any OS and with any IDE. However, you must make sure that your solution (code) is working using the provided running scripts and testcases. If the running scripts or the testcases fail the submission will be normally be considered unsuccessful.

Note : Students who want to install anaconda should have complete knowledge about it and only then set up conda environments in their system, otherwise we highly recommend to use system python and the steps suggested below.

Recommended version numbers: python : 3.6+
OpenCV : 4.1+

2.1 Install Python

Linux

```
sudo apt install python3
```

Windows
Tutorial
Doc

2.2 Setup Virtual Environment

Linux/Mac users

```
cd skeleton
python3 -m venv venv
source venv/bin/activate
pip install opencv-python numpy pytest
```

Windows users

```
cd skeleton
python -m venv venv
```

```
.\venv\Scripts\Activate.ps1
pip install opencv-python numpy pytest
```

2.3 Directory Structure

The project directory structure will be persistent across exercises. Submissions are expected to contain a snapshot of the project directory.

```
ComputerVision2021
├── ex0
│   ├── __init__.py
│   └── functions.py
├── ex1
│   ├── __init__.py
│   └── ...
├── ...
├── docs
│   ├── ex0.pdf
│   ├── ex1.pdf
│   └── ...
├── data
│   ├── ex0
│   │   └── img.png
│   ├── ex1
│   └── ...
├── bin
│   ├── run_ex0.py
│   ├── run_ex1.py
│   └── ...
└── test
    ├── test_ex0.py
    ├── test_ex1.py
    └── ...
```

- Each exercise will consist of writing a python module. All team members are expected to update the __authors__ dictionary with their fauId and their names in __init__.py
- Each exercise will have it's own launching script found in ./bin. These scripts will usually be provided completed and demonstrate how the implemented module is given.
- Each exercise will have it's own pytest testcase found in ./test. Passing these test-cases will normally mean that the exercise is successful. Although you are allowed and encouraged to write you own test-cases, the original test cases should not be changed. Test cases passing are a good indication of a correct submission.
- Each exercise has it's announcement in docs/ex0.pdf, docs/ex1.pdf, ...

2.4 Run Code

```
# Go to exercise directory
cd skeleton
# Activate virtual environment
source venv/bin/activate [LINUX/MacOS]
.\venv\Scripts\Activate.ps1 [WINDOWS]
# Execute
python3 bin/run_ex0.py
# Test
pytest test/test_ex0.py
```

3 OpenCV Image Processing

To get used to the most basic OpenCV functions implement the following in `ex0/functions.py`:

3.1 Image Loading and Saving

Implement `show_images` and `save_images` in `ex0/functions.py`. Load the image `img.png` and display it on screen. Use the OpenCV functions `imread` and `imshow`. If your program exits, all created windows will close. Use the function `waitKey` to stall the program once all windows have been showed until a key has been pressed.

Similar to the loading, you can save the image by calling the function `imwrite`. Save the image as `img.jpg`

3.2 Resizing

Implement `scale_down` in `ex0/functions.py`. Resize the image by a factor of 0.5 in both directions with the OpenCV function `resize`. Show the resized image on screen and save it as `small.png`.

3.3 Color Channels

Implement `separate_channels` in `ex0/functions.py`. Create three images, one for each channel (red, green, blue). Make sure these objects have the same size and type as the input image. Iterate over every pixel of the input image and store each channel individually in one of the 3 images. A single row of an image is accessed in the following way:

```
// Set the blue channel of the fifth image column to 0
// Note: OpenCV stores images in BGR format.
img[i, 4, 0] = 0
```

Hint: An efficient solution has no loops in python/numpy.

Display the three images on screen. They should look like this:

Figure 1: The red, green, and blue channel of the input image.