Darius Rückert
Prathmesh Madhu
23.04.2020

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. Passing 50% of the exercises (starting with Exercise 1) qualifies you for the oral exam.
- There will be an extra exercise sheet on convolutional neural networks. Details follow later.
- You can gain up to 20 points per exercise sheet. There are 100 points in total, therefore you need at least 50 points to pass the exercises.
- 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 in groups of 2 or 3.
- 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.

[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 default build scripts. If your code crashes, you will be granted 0 points.

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
```

Windows users

```
cd skeleton
python -m venv venv
.\venv\Scripts\Activate.ps1
pip install opencv-python numpy
```

2.3 Run Code

```
# Go to exercise directory
cd skeleton
# Activate virtual environment
source venv/bin/activate [LINUX/MacOS]
.\venv\Scripts\Activate.ps1 [WINDOWS]
# Execute
python src/main.py
```

3 OpenCV Image Processing

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

3.1 Image Loading and Saving

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 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

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

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 pixel of an image is accessed in the following way:

```
// Set the blue channel of pixel (i,j) to 0
// Note: OpenCV stores images in BGR format.
img[i,j,0] = 0;
```

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

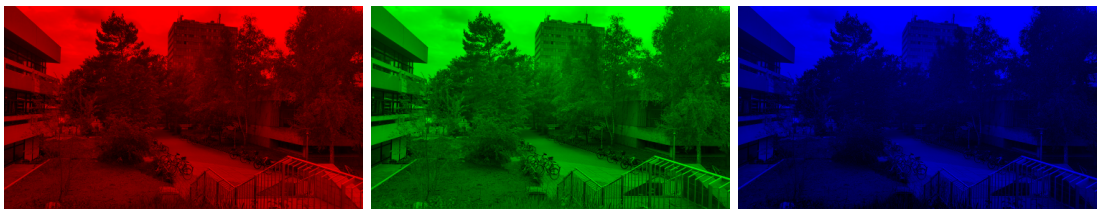

Figure 1: The red, green, and blue channel of the input image.