Darius Rückert
AmirAbbas Davari
16.04.2018

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.

The exercises are structured as follows:

- There are 7 exercise sheets. Passing 50% of the exercises (starting with Exercise 1) qualifies you for the oral exam.
- You can gain up to 15 points per exercise sheet (except Exercise 7, which gives only 10 points). There are 100 points in total, therefore you need at least 50 points to pass the exercises.
- Exercises 5,6, and 7 depend on each other. You need to complete exercise 5 before you can start exercise 6 (same with 7 and 6).
- The exercise sheets are going to be released on StudOn on Monday.
- 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 Compile and Run

The exercises are written in C++ and use the open-source library OpenCV. The build files will be generated by cmake. 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 on the Huber-CIP pool computers. If your code does not work in Huber-CIP you will be granted 0 points.

2.1 In Huber-CIP

In Huber-CIP, everything is already installed and you can get the code running by typing the following in a terminal:

```
#navigate to the exercise directory
cd <path_to_exercise>/Ex0
#generate unix makefiles
cmake .
#compile and link
make
#run
./CV
```

2.2 At Home (Linux)

Install OpenCV and cmake with your package manager. On a debian based system type:

```
sudo apt-get install libopencv-dev
sudo apt-get install cmake
```

After that, proceed as if you were in Huber-CIP.

2.3 At Home (Windows)

- Install OpenCV by following the instructions at https://docs.opencv.org/2.4/doc/tutorials/introduction/windows_install/windows_install.html.
- Install cmake.
- Start cmake and set the source and build directory to the exercise folder.
- Generate the Visual Studio project.

3 OpenCV Image Processing

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

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 `Mat` objects, 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:

```
Mat img = ...;

// Read the pixel at position (10,10) and store it in e.
Vec3b e = img.at<Vec3b>(10,10);
// Note: OpenCV images are stored in BGR order!
// Set the blue channel to 0.
e[0] = 0;
// Store the modified pixel back at position (10,10)
img.at<Vec3b>(10,10) = e;
```

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

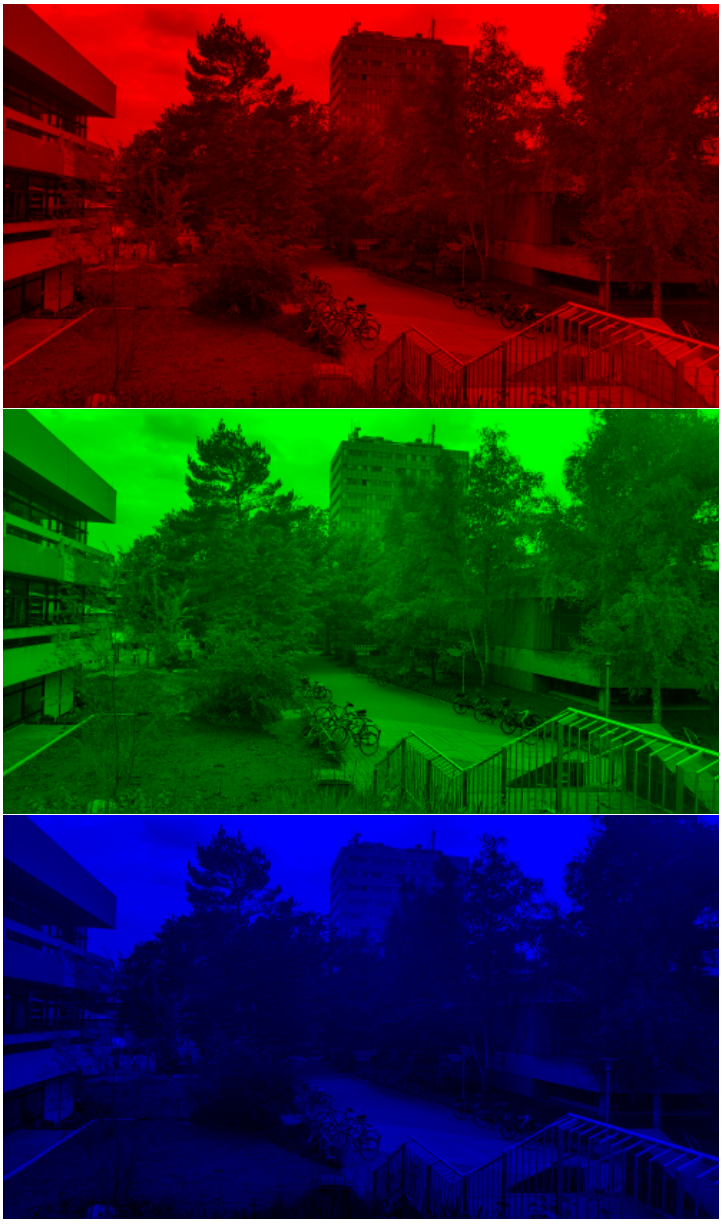

Figure 1: The red, green, and blue channel of the input image.