
Heidelberg University

Visual Learning Lab

# Exercises for 3D Computer Vision

Titus Leistner
Philip Grassal
Raphael Baumgartner
Prof. Dr. Carsten Rother

November 26, 2020

# Contents

# Introduction

Welcome to the online exercise for Computer Vision: 3D Reconstruction. We prepared six tasks to introduce you to the practical aspects of Computer Vision and prepare you for the mini projects at the end of the semester. The exercises will cover the following aspects:

- The Numpy and Matplotlib modules which are essential for any scientific computation project with Python
- Image processing on the example of fast filtering and corner detection
- Feature descriptors and Panorama stitching
- PyTorch, tensors and Autograd
- Different neural network architectures for image classification on MNIST
- Differentiable RANSAC as an example for a recent research topic in our group

**Please note:** We are aware that many of you already have experience with some of those topics. However, we also received many requests from students with very little or no programming experience. Therefore, we made the decision to also include some basic tasks, especially for the first exercise. We made the first exercise optional, so feel free to skip it, if you are already familiar with Python and Numpy.

## Admin

We will publish the exercises on GitHub [1]. Please note, that we might update the repository as well as this document from time to time if we encounter some bugs in the code or mistakes in the tasks. The repository contains installation information and one iPython-notebook (.ipynb-file) per exercise. In the notebooks, you will find some already implemented code and more detailed information about the specific tasks. We assigned a score to each subtask. Please note that you have to score at least 50% of all points for each exercise in order to work on a mini project and get a mark for this lecture.

Each exercise has to be submitted via Moodle before the assigned deadline. Please do not upload more than the .ipynb-file itself. If you have any comments for the corrector regarding your solution, add a new markdown-cell inside the notebook. Please use the forums on Moodle for all your questions. We will create a new forum for each exercise. Questions regarding the installation and setup of the repository can be asked in the forum for the first exercise. You should use the Admin forum for all other admin questions.

You will receive feedback, but no points for the first optional exercise. All other exercises can be edited in groups. There will be a "Groups" forum to find group partners. We will then switch the Moodle exercises to group modes, hence, you only have to submit one solution per group.

Last but not least we want to encourage you to also present unconventional and creative solutions, play around with parameters and do some of the optional tasks.

But most importantly: Have fun, stay healthy and happy despite this difficult time!

Titus Leistner, Philip Grassal, Raphael Baumgartner and Carsten Rother

# 1. Scientific Python

This exercise gives you a short introduction into NumPy [2], which is widely used for numerical computation with Python, and Matplotlib [3] for plotting graphs and presenting your results. As this only covers basic preliminaries and many of you should already be familiar with it, it is not mandatory to hand in this exercise.

## Setup

The first task is to setup our repository [1]. You find detailed installation information in the README.md. We tested our setup on Linux (Ubuntu and Arch), MacOS and Windows 10. If you encounter (or solve) any problem regarding the setup, please open a new thread in the forum for the first exercise.

## 1.1 NumPy

NumPy forms the basis of the Python scientific stack. Its main component is the `numpy.ndarray` class for $n$-dimensional arrays. Because it is mostly implemented in Fortran, computations using NumPy arrays are way faster than e.g. operations on Python lists. Let's take a look at a minimal example:

```
1 # import the numpy module
2 # note that np is the common naming convention
3 import numpy as np
4
5 # create a new 1D array with 4 elements from a Python list
6 a = np.array([0, 0.32, 10, 12], dtype=np.float)
7
8 # perform a basic operation with a floating-point scalar
9 b = a * 42.0
10
11 # use slicing to get the last two elements of this array
12 c = b[-2:]
```

Your task is to get used to scalar, vector and matrix operations with NumPy.

## 1.2 Scikit-Image

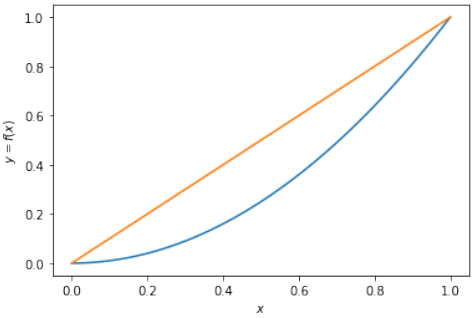
For all exercises we will use scikit-image for basic image loading, storing and manipulation. This module is part of the SciPy ecosystem for scientific computing and rather light-weight and easy to install compared to e.g. OpenCV.

## 1.3 Matplotlib

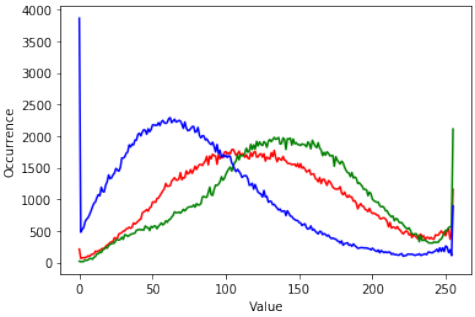
Matplotlib is an extremely powerful library for scientific visualization. In this exercise you will use it to output your processed images as well as some simple function plots and histograms. Let's again look at some minimal example:

```
# imports
# again, note the naming conventions for np and plt
import numpy as np
import matplotlib.pyplot as plt

# create an array with numbers between 0 and 1
xs = np.linspace(0, 1)

# compute an array for y = f(x) = x^2
ys = xs ** 2.0

# create a new plot for our x^2 function
plt.plot(xs, ys)

# add another plot for f(x) = x
plt.plot(xs, xs)

# define some axis labels
plt.xlabel('$x$')
plt.ylabel('$y = f(x)$')

# present your plot
plt.show()
```

This little code snippet creates the following plot:



Your task is to plot the color histogram of a previously loaded image. To create the histogram data, you are allowed to use an `skimage` helper function. Your result should look similar to this plot:




## 1.4 HSV Color Space Conversion

The last task of this exercise deals with different color spaces. A color space defines how the color components of an image are stored and processed. Due to its usage in most screens, the RGB color space is the most common and widespread one. However, several other color spaces also play an important role in Computer Vision. E.g. the LAB space is used by many algorithms for image colorization. The HSV space, however, is especially useful for data augmentation. Data augmentation techniques improve the generalization of Machine Learning models without the need for more training data. This is achieved by modification of the existing data. A common data augmentation technique for Computer Vision is the hue rotation of an image. While this is hard to achieve in RGB space, it is trivial in HSV space as the hue is a separate component. Your task is to implement the conversion from RGB to HSV and back, by using Numpy operations only (no scikit-image or other modules allowed).

### 1.4.1 Convert RGB to HSV

We first compute the value $V$ which is defined as the maximum component in RGB space

$$X_{\max} = \max(R, G, B) = V. \tag{1.1}$$

With the minimal component

$$X_{\min} = \min(R, G, B) \tag{1.2}$$

we can compute the range which is also called chroma

$$C = X_{\max} - X_{\min}, \tag{1.3}$$

as well as the mid range, also called luminance

$$L = \frac{X_{\max} + X_{\min}}{2}. \tag{1.4}$$

The hue component is defined on a full circle. A 360° hue rotation travels through the whole visible color spectrum. We therefore have to find the closest component in RGB space in order to decide for an 120° segment of the circle:

$$H = \begin{cases} 0, & \text{if } C = 0 \\ 60°(0 + \frac{G-B}{C}), & \text{if } V = R \\ 60°(2 + \frac{B-R}{C}), & \text{if } V = G \\ 60°(4 + \frac{R-G}{C}), & \text{if } V = B \end{cases} \tag{1.5}$$

We finally compute the inverse proportion of white, called saturation

$$S = \begin{cases} 0, & \text{if } V = 0 \\ \frac{C}{V}, & \text{otherwise} \end{cases}. \tag{1.6}$$

### 1.4.2 Convert HSV back to RGB

The opposite direction can be achieved by first computing chroma

$$C = V \times S \tag{1.7}$$

and dividing the 360° spectrum into its components

$$H' = \frac{H}{60°}, \tag{1.8}$$

$$X = C \times (1 - \mid H' \mod 2 - 1 \mid). \tag{1.9}$$

We then compute the "pure" RGB components

$$(R_1, G_1, B_1) = \begin{cases} (0,0,0), & \text{if } H \text{ is undefined} \\ (C,X,0), & \text{if } 0 \leq H' \leq 1 \\ (X,C,0), & \text{if } 1 < H' \leq 2 \\ (0,C,X), & \text{if } 2 < H' \leq 3 \\ (0,X,C), & \text{if } 3 < H' \leq 4 \\ (X,0,C), & \text{if } 4 < H' \leq 5 \\ (C,0,X), & \text{if } 5 < H' \leq 6 \end{cases} \tag{1.10}$$

and finally add the white proportion

$$m = V - C \tag{1.11}$$

to each component

$$(R, G, B) = (R_1 + m, G_1 + m, B_1 + m). \tag{1.12}$$

Your last subtask is to also plot the color histogram for the hue-rotated image. A shift of the color components should be clearly visible.

# Bibliography


[1] T. Leistner, R. Baumgartner, and C. Rother, *Exercises for 3D Computer Vision*. [Online]. Available: https://github.com/titus-leistner/3dcv-students

[2] *NumPy*. [Online]. Available: https://numpy.org

[3] *Matplotlib*. [Online]. Available: https://matplotlib.org

[4] *Scikit-Image*. [Online]. Available: https://scikit-image.org

[5] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *NEURIPS 32*, 2019.

[6] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, 2017.

[7] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother, "DSAC - differentiable RANSAC for camera localization," in *CVPR*, 2017.

[8] *Least-Squares Circle Fit - DTC Documentation*. [Online]. Available: https://dtcenter.org/met/users/docs/write_ups/circle_fit.pdf