DARMSTADT UNIVERSITY
B. FRÖMMER, E. HERGENRÖTHER, B. MEYER

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

4. JUNE 2023

VISUAL COMPUTING SOSE 2023
TASK 4

The aim of this task sheet is to familiarise yourself with the OpenCV library. OpenCV provides the basics for working with images as well as many functions used in image processing and -analysis are frequently used. It is available as an open source project at: http://opencv.org/

4.1 Installing OpenCV

Detailed installation instructions for C++ are available in Moodle.
You can find good documentation in the book Learning OpenCV by Gary Bradski and Adrian Kaehler or here: https://docs.opencv.org/4.7.0/d1/dfb/intro.html

(Recommended) Alternative: Installing OpenCV under C++ can be a bit tedious. It is much easier to integrate OpenCV into a Python project:

a) Install a Python IDE of your choice (for example PyCharm Community by JetBrains, free as a student licence at https://www.jetbrains.com/pycharm/download/#section= windows).

b) Install a current Python version (at https://www.python.org/downloads/, make sure that the PATH variable is set accordingly during installation)

c) Create a Python project in the IDE as a test. In PyCharm, you can use *python --version in* the terminal to display the installed Python version number. If this does not work, your Python environment is not found.

d) Installing additional packages: Python comes with an integrated package wizard, PIP (Package Installer for Python). If you would like to manage different projects with different Python versions, we recommend the package manager *Anaconda* (not required for this internship).

e) With pip install opencv-python you automatically install the current version of OpenCV. More information is available at https://pypi.org/project/opencv-python/.

f) Optional: The editing of read-in image data can be simplified with the help of the maths library Numpy. pip install numpy installs the library accordingly.

Copy the image yoshi.png into your project folder and finally test your installation by adding the following lines :

```
# import numpy as np ## optional import
cv2
print("OpenCV version: " + cv2. version ) img =
cv2.imread("yoshi.png") cv2.imshow("image",
img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

4.2 First steps

First, let's look at simple OpenCV functions for image manipulation and display. Use known functionalities or use online resources such as the documentation of the OpenCV library to:

- to load the attached "yoshi.png".
- output the width, height and number of colour channels on the console.
- change the image data format to floating point.
- display the image until a key is pressed (both as uint8 and as float).
- draw a red square with 10x10 pixels in the middle of the picture.
- replace every 5th line with black pixels.
- save the image to the hard disk.

4.3 Colour spaces

In this task we want to deal with colour space conversions.

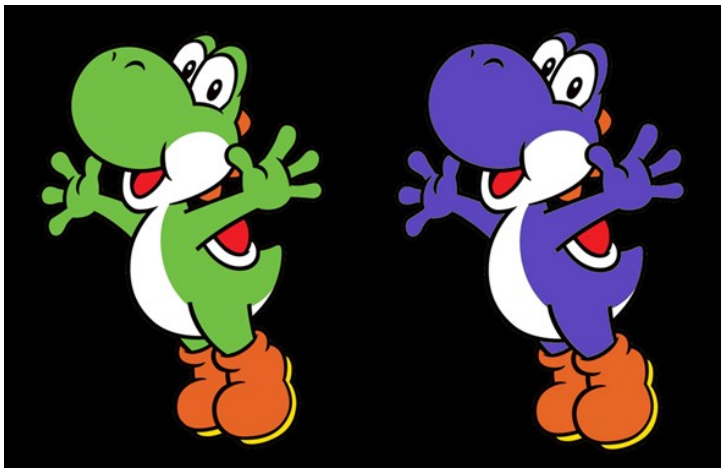
(1) In addition to Yoshi, load the corresponding mask "mask.png" into your programme.

(2) Transfer the Yoshi image to the HSV colour space.

(3) Change the H-value in the Yoshi image for all white pixels in the mask image.

(4) Display the new image on the screen (but note that imshow() can only process BGR images correctly). The result should look as shown.

(5) Optional: Build a slider to manually set the H-value to any value and update the displayed image.



4.4 Reinhard's Colour Transfer

In this exercise you will implement a variant of Reinhard's colour transfer. (Source: Color Transfer between images by Reinhard, Ashikmin, Gooch and Shirley, available in Moodle). The idea of colour transfer is to adapt the basic colour scheme of an image (here: input) to a specific target colour scheme (here: target). An example is shown in Fig. 1.
Proceed as follows:

(1) Load two images as "Input" and "Target" and convert them into floating point values (in the range of 0 to 1).

(2) Convert both images to the Lab colour space.

(3) For each colour channel (separately):

i) Subtract the mean value of the input from the input (the new mean value of the input becomes 0) .

ii) Divide the input by its standard deviation (the new standard deviation becomes 1).

iii) Multiply the input by the standard deviation of the target.

iv) Add the mean value of the target to the input.

(4) Convert the input back to BGR.

Also test the algorithm with the RGB and HSV colour spaces you know and other images. Which colour spaces are suitable for the algorithm, which are not? With which images does the colour transformation work well? Can you explain why?

Figure 1: **Left: Input image with light blue sky. Middle: Target image with desired colouring. Right: Adjusted input image with new colouring.**