Prerequisite test of the Computer Vision course at the University of Helsinki in May 2018

Konsta Kutvonen

May 14, 2018

1 Introduction

The purpose of this document and the prerequisite test for the computer vision course is, first, to verify that all participants have a functioning software setup in their computers when the course starts. Second, it makes sure that the participants are able to compose the results of their home works into PDF reports and to submit them to the lecturer.

In general, the results of this prerequisite test and the daily home work reports can be written by using also any other software than pdfLaTeX. The only thing that matters is that the submitted report is in a single PDF file and contains the used program codes and generated images in a readable and clear form.

The aim is, that you compile and run the two included programs `test.cc` (written in C++) and `test.py` (written in Python). Running them successfully will produce four images that will be included in the prerequisite report that you email to the lecturer.

If you encounter technical problems in completing the prerequisite test, you can contact the course teaching assistant Gür Ersalan by phone 041 807 0887 or by email `muzaffer.ersalan@helsinki.fi` and ask support from him.

Deadline for the submission is Friday 11.5.2018 at 23:59 o'clock!

2 Instructions

2.1 Copy the report template

The distributed ZIP file contains this file in both PDF and pdfLaTeX source formats. The source code of the document is stored in `your_name_report_0.tex`. Copy it to a new file whose name contains your own name. (The daily home work reports you can create from this same example by replacing the "0" with numbers "1"... "4".)

Read what follows in this document and modify it by answering to the stated questions and by doing the other requested updates!

2.2 Fill in the statistics

First some statistics. Answer to the following questions by replacing the lecturer's answers with your own:

- What is your operating system? *Mac OS*
- What is its distribution and/or release? *High Sierra 10.13.3*
- Have you used Python before? *Yes*
- Have you used C++ before? *No*
- Have you used OpenCV before? *No*
- Have you taken a course in image processing or computer vision? *No*

2.3 Run `test.cc`

You should be able to run `test.cc` by first compiling and linking it with the GNU C++ compiler `g++` and then executing the resulting *a.out* executable.

This is an example of command lines to perform the compilation and to run the created executable:

```
g++ test.cc -lopencv_core -lopencv_imgproc -lopencv_highgui -lopencv_imgcodecs -lopencv_plot
./a.out
```

At least the following software packages need to be installed for the compilation and linking to succeed:

- g++
- libopencv-core-dev
- libopencv-highgui-dev
- libopencv-imgproc-dev
- libopencv-imgcodecs-dev
- I installed OpenCV with Brew and got the necessary libraries.

Complete the above list with the possible extra packages you needed to install to get the code run.

Note: The OpenCV class `cv::plot::Plot2d` has changed slightly between OpenCV releases 3.1.0 and 3.4.1. The code in `test.cc` expects the version is 3.1.0, but the newer interface can be taken into use by changing the out-commented lines.

Running the code will first show a color image of a football player. Pressing a key while input focus is in the image or destroying the window will show a related greyscale image. Finally you should see a yellow plot on black background. Closing that plot will end the run.

During the run you should have seen the following output:

```
x.y.z
rows=342 cols=548 channels=3 type=16 (type 16 means CV_8UC3)
rows=342 cols=548 channels=1 type=0 (type 0 means CV_8UC1)
```

Also, two new image files `green.jpg` and `projection.png` should have appeared.

Answer to the following questions:

- What was `x.y.z` (OpenCV version) in your output? *3.4.1*
- Did you see the two images and one plot? *Yes*

2.4 Run `test.py`

You should be able to run `test.py` by executing it from Linux's command line or by some other similar means. At least the following software packages need to be installed for it to work:

- python3
- python3-opencv

Complete the above list with the possible extra packages you needed to install to get the code run.

Running the code will first show a color image of a football player. Pressing a key while input focus is in the image or destroying the window will show the same image in greyscale. When that image is closed, you should see an image which is partly color and partly greyscale. Finally you should see a histogram of the greyscale intensity values. (If you don't remember what a histogram is, find it out. . .) Closing that plot will end the run.

During the run you should have seen the following output:

```
x.y.z
<class 'numpy.ndarray'> uint8 (342, 548, 3)
<class 'numpy.ndarray'> uint8 (342, 548)
```

Also, two new image files `grass.jpg` and `histogram.png` should have emerged.

Answer to the following questions:

- What was `x.y.z` (OpenCV version) in your output? *3.4.0*
- Did you see the three images and one plot? *Yes*
- How the third image was partly color and partly greyscale? *It was greyscale except for the grass*
- Do you now know what is a histogram? *Yes*

2.5 pdfLaTeXing the report

You'll need to have some sort of TeX distribution installed to compile the modified version of this document to a PDF file. Running pdfLaTeX is done by running twice:

```
pdflatex your_name_report_0
```

At least the following software packages need to be installed for the pdfLaTeXing to succeed:

- texlive-latex-base
- *Installed MacTex and texmaker*

3 Code example

Source code can be included in the document inside a `verbatim` environment. Let's test it now because we'll need it later. . .

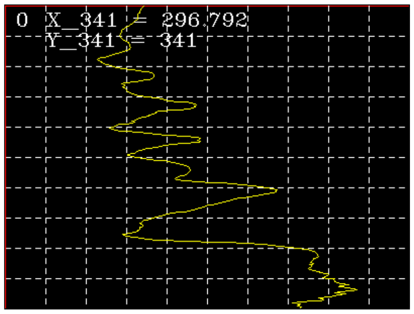
```
#! /usr/bin/env python3

# https://docs.opencv.org/3.1.0/d6/d00/tutorial_py_root.html
# https://github.com/abidrahmank/OpenCV2-Python-Tutorials

import cv2
import numpy as np
import matplotlib.pyplot as plt

print(cv2.__version__)

#flags = [i for i in dir(cv2) if i.startswith('COLOR_')]
#print(flags)

img_bgr = cv2.imread('messi5.jpg')
print(type(img_bgr), img_bgr.dtype, img_bgr.shape)
cv2.imshow('BGR', img_bgr)
cv2.waitKey(0)

img_g = cv2.cvtColor(img_bgr, cv2.COLOR_BGR2GRAY)
print(type(img_g), img_bgr.dtype, img_g.shape)
cv2.imshow('grey', img_g)
cv2.waitKey(0)

z = img_bgr
for i in range(z.shape[0]):
    for j in range(z.shape[1]):
        if 1.2*z[i,j,0]>z[i,j,1] or 1.2*z[i,j,2]>z[i,j,1]:
            z[i,j] = [img_g[i,j]]*3

cv2.imwrite('grass.jpg', z)
cv2.imshow('grass', z)
cv2.waitKey(0)

h = [0]*256
for i in img_g.flatten():
    h[i] += 1

plt.plot(h)
plt.savefig('histogram.png')
plt.show()
```

It's a good practice to keep the code lines shorter than 80 characters per line in order to make them fit on the page!

4 Image examples

Images can be included in pdfLaTeX documents by using the `includegraphics` command.

Running the `test.cc` and `test.py` programs should produce the following four images.



5 Submitting the report

When you have answered to the questions in Section 2, inserted the code in Section 3 and verified that the four generated images are visible in Section 4, send the PDF by email to `jorma.laaksonen@aalto.fi`

Welcome to the course starting 14.5.2018 at 9:15 o'clock in lecture hall Exactum B222!