IMPR 2017

Exercise 0

Due date: Nov. 9, 2017, 11:55pm.

Note, this exercise is not mandatory and won't be graded

This exercise can be submitted in pairs.

The goal of this exercise is to get hands-on experience in working with images and numerical computation using the Python environment.

You are not allowed to use opencv built in functions for the tasks in this exercise except reading images from disk. Only numpy functions are allowed.

Software installation:

1. Install python environment: https://www.anaconda.com/download/

   Note that in the course we are working with python 3.x and not with python 2.7

2. Install openCV whl on your python environment:

   https://pypi.python.org/pypi/opencv-python

General programming in python:

1. We will not cover in the course general aspects of programming in Python. For those who are not familiar with python we recommend to go over chapters 1-4 (at least) in the following tutorial:

   https://docs.python.org/3/tutorial/

   Note that exercises in the IMPR course will focus on the numerical computations rather than on general programming in Python.

Numerical computation in Python:

Python has an optimized library for numerical computations called numpy.

To get familiar with this library please go over the following tutorial: https://docs.scipy.org/doc/numpy-dev/user/quickstart.html

Task 1: Some numerical programming in python

1.a. Create a function that takes an integer N as input, and generate a N by N matrix filled with random numbers sampled from the uniform distribution in the range (0,1].

The function should return the matrix and the row, column index of the minimum and maximum values along with the minimum and maximum values

1.b. Create a function that takes a Nx2 array of 2D points in Cartesian coordinates and return a Nx2 array of the same points in polar coordinates. You should use Radians to represent the angle in the Polar coordinate system)

Cartesian coordinates: https://en.wikipedia.org/wiki/Cartesian_coordinate_system

Polar coordinates and conversion from Cartesian coordinates: https://en.wikipedia.org/wiki/Polar_coordinate_system

Task 2: RGB to Grayscale conversion

In this task you should write a function that takes as input a color image (RGB) and the RGB 2 Gray conversion method, and return the converted grayscale image.

Conversion methods are as follows:

Lightness: (max(R, G, B) + min(R, G, B)) / 2.

Average: (R + G + B) / 3

Luminosity: 0.21 R + 0.72 G + 0.07 B   #Default method

Note that opencv return the image in the BGR order rather than in RGB.

Task 3: RGB to YIQ color space conversion:

3.a. Write a function that transform an RGB image into the YIQ colorspace.

The function takes as input a 3D matrix [height x width x color channels] with the 3rd dimension is the RGB channels and return a 3D matrix [height x width x color channels] with the 3rd dimension as the YIQ channels.

The YIQ colorspace: https://en.wikipedia.org/wiki/YIQ

The transformation between the color space is given by:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

3.b. Write a function that transform a YIQ image into the RGB colorspace.

Files included:

ex0Driver.py: an example that defines and demonstrates the API for the functions you should write

./Images/* :set of images to be used with the driver

You should submit:

Ex0.zip file contains:

1. ex0.py: the code of your solution. The file ex0driver.py should run with your ex0.py without any error or warning.

2. readme: include your names

Submission instructions:

Through the course moodle website