Assignment 1: Python Introduction

Due Date: 06 October 2023, 23:59

Introduction

The goal of this assignment is to familiarize yourself with the Python libraries NumPy, Matplotlib and OpenCV. Most of the questions can be implemented in a few lines of code. The technical skills acquired by this assignment will be useful for the future lab exercises.

1 Basic Matrix/Vector Manipulation (50 points)

Print your results to the console for all questions. For questions (e) and (f) you should not use any built-in functions that directly compute the expected result.

(a) Define matrix $\boldsymbol{M}$ and vectors $\boldsymbol{a}$,$\boldsymbol{b}$ and $\boldsymbol{c}$ in Python (hint: use *numpy*).

$$\boldsymbol{M} = \begin{bmatrix} 4 & 8 & 11 \\ 3 & 6 & 2 \\ 9 & 7 & 1 \\ 10 & 0 & 5 \end{bmatrix}, \boldsymbol{a} = \begin{bmatrix} 3 \\ 1 \\ 4 \end{bmatrix}, \boldsymbol{b} = \begin{bmatrix} -2 \\ 6 \\ 0 \end{bmatrix} \quad \boldsymbol{c} = \begin{bmatrix} 7 \\ 3 \\ 6 \end{bmatrix}$$

(b) Calculate the dot product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

(c) Calculate the element-wise product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

(d) Calculate $\boldsymbol{Ma}(\boldsymbol{c}^\intercal \boldsymbol{b})$.

(e) Find the magnitude of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

(f) Normalize vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

(g) Find the angle between vectors $\boldsymbol{a}$ and $\boldsymbol{c}$ (in degrees).

(h) Find vector $\boldsymbol{d}$ that is perpendicular to vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

(i) Use vectors $\boldsymbol{a}$, $\boldsymbol{b}$ and $\boldsymbol{d}$ to create an orthonormal basis. Verify your results e.g. check that the produced matrix is indeed orthonormal (hint: *numpy.isclose()*, *numpy.allclose()*, *assert()*, *check the properties of an orthonormal basis*).

2 Basic Image Manipulations (50 points)

Plot your results using matplotlib. For questions (b) - (f) you should not use any built-in functions that directly compute the expected result. For the last question (f) your results should be in a 100% agreement with OpenCV's under a threshold of 1, i.e., the value of each channel of each pixel between the two resized images should differ at most by 1.

(a) Load images *image1.jpg* and *image2.jpg*. Retain their initial color space. Plot the two images in a 1×2 grid.

(b) Convert the images to double precision and rescale them to stretch from minimum value 0 to maximum value 1. Plot the two images in a 1×2 grid.

(c) Add the images together and re-normalize them to have minimum value 0 and maximum value 1 (plot the output image).

(d) Create a new image such that the left half of the image is the left half of *image2* and the right half of the image is the right half of *image1* (plot the output image).

(e) Without using a for-loop, create a new image such that every even row is the corresponding row from *image1* and every odd row is the corresponding row from *image2* (plot the output image).

(f) Resize *image3.png* by 5 i.e. increase its width and height by 5, using bilinear interpolation. In cases near edges or corners use only the available neighboring pixels from the original image (no padding). Verify your results using *cv2.resize(..., interpolation=cv2.INTER_LINEAR)*. Plot the three images (original, OpenCV resized image and the resized image from your implementation) in a 1×3 grid. For the pixel agreement check between yours and OpenCV's results you can use the following function:

```python
def check_resizing(img1, img2, value_thresh=1):
    """
        Check if pixel values agree for img1 and img2 under the given threshold
    :param img1: numpy.array(uint8), input resized img1
    :param img2: numpy.array(uint8), input resized img2
    :param value_thresh: int, pixel value threshold
    :return:
        None
    """

    assert img1.shape == img2.shape
    img1_reshaped = img1.reshape(-1, 3).astype(np.int32)
    img2_reshaped = img2.reshape(-1, 3).astype(np.int32)
    dist = np.abs(img1_reshaped - img2_reshaped)
    n_pixels = np.prod([s for s in img1.shape[:2]])
    dist_per_pixel = np.sum(dist, axis=1)
    n_correct_pixels = np.sum(dist_per_pixel <= img1.shape[-1] * value_thresh)
    pixel_ratio = n_correct_pixels / n_pixels
    print(f'Resizing: {pixel_ratio*100:.1f}% pixels in agreement (thr={value_thresh})')
```

3 Extra Credits: Non-symmetric resizing (5 points)

For this **extra credits** question, you will need to revisit question (2.f) and resize the given image by 160 along the x-axis (width) and by 90 along the y-axis (height). Again, your results should be in a 100% agreement with OpenCV's under a threshold of 1.

4 Instructions

- The assignment is due by **Friday October 6th, at 23:59** (see the course's GitHub repository for more details regarding the late assignments policy).
- The assignment should be submitted **only** through Moodle. Compress all the needed **source code** files, e.g. all *.py files into a .zip file and name it as follows "**Lab_Assignment_1_<ID-Number>.zip**".
- For further questions you can either create an issue on the course's GitHub repository or contact me via email mloizo11@ucy.ac.cy.
- All lab assignments should be conducted **independently**.
- The Moss system will be used to detect code similarity.
- Plagiarism and/or cheating will be punished with a grade of zero for the current assignment. A second offense will carry additional penalties.