



Assignment 1: Python Introduction

Due Date: 17 February 2025, 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 \mathbf{M} and vectors \mathbf{a} , \mathbf{b} and \mathbf{c} in Python (hint: use *numpy*).

$$\mathbf{M} = \begin{bmatrix} 2 & 6 & 3 \\ 17 & 2 & 17 \\ 12 & 2 & 6 \\ 17 & 4 & 14 \\ 0 & 17 & 18 \end{bmatrix}, \mathbf{a} = \begin{bmatrix} 3 \\ 0 \\ -5 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 5 \\ 0 \\ 3 \end{bmatrix}, \mathbf{c} = \begin{bmatrix} 7 \\ 13 \\ 7 \end{bmatrix}$$

- (b) Calculate the dot product of vectors \mathbf{b} and \mathbf{c} .
- (c) Calculate the element-wise product of matrix \mathbf{M} and vector \mathbf{c} .
- (d) Calculate $\mathbf{M}\mathbf{b}(\mathbf{a}^T\mathbf{c})$.
- (e) Find the magnitude of vectors \mathbf{a} , \mathbf{b} and \mathbf{c} .
- (f) Normalize vectors \mathbf{a} , \mathbf{b} and \mathbf{c} .
- (g) Find the angle between vectors \mathbf{a} and \mathbf{b} (in degrees).
- (h) Calculate the vector \mathbf{d} that is perpendicular to vectors \mathbf{a} and \mathbf{b} .
- (i) Use vectors \mathbf{a} , \mathbf{b} and \mathbf{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)

For this part of the assignment, begin by downloading the *Assignment_1_material.zip* file from the Assignment 1 section on Moodle. Once downloaded, extract the contents to your working directory. Make sure that the images *image1.jpg*, *image2.jpg* and *image3.png* have been successfully extracted. For plotting your results, you will need to use *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 top half of the image is the bottom half of *image2* and the bottom half of the image is the top half of *image1* (plot the output image).
- (e) Without using a for-loop, create a new image such that every even column is the corresponding column from *image2* and every odd column is the corresponding column from *image1* (plot the output image).
- (f) Resize *image3.png* by 6 i.e. increase its width and height by a factor of 6, 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:

```
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 a factor of 38 along the x-axis (width) and by a factor of 22 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 **Monday February 17th, at 23:59** (see the course's [GitHub repository](#) for more details regarding the late assignment 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_<UC-ID-Number>.zip**".
- For further questions you can create an [issue](#) on the course's GitHub repository.
- 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.