Exercise Sheet 0(b)*

Computer Vision 1
WiSe2021

Issue date: 06. Nov. 2020 - **Not to be sumbitted**

Exercise 1 — NumPy and Matplotlib Tutorial - 0 Points - Theoretical exercise

Go through the short tutorial[1] from Justin Johnson and Co. to NumPy. Check only the parts for NumPy (up to Numpy Documentation) and Matplotlib (up to the end). Then test your knowledge and answer the following questions in writing (each answer is 1-2 sentences).

1. How can the number of rows and columns be determined in a NumPy array?
2. What is the difference between the functions `numpy.array` and `numpy.zeros`?
3. How can the data type of a given NumPy array be determined?
4. What does `.T` mean behind the variable name of a NumPy array? For example: `a.T`
5. What do the functions `xlabel` and `ylabel` from `matplotlib.pyplot` do?

Exercise 2 — First steps in NumPy - 0 Points - Programming task

In this exercise, matrices and vectors from NumPy arrays are to be generated and manipulated. All steps should be programmed in Python and recorded in a Python script. Only NumPy is permitted for this task.

1. First create a vector `u` as a 1D array with 100 zeros.
2. Also create a vector `v` as a 1D array from the list: `[0,1,2,3,4,5,6,7,8,9,10,11]`.
3. Transform the vector `v` into a matrix `m` (2D array), which consists of three rows and four columns. Hint: NumPy has a function to change the shape of an array.
4. Multiply all entries of the matrix `m` with the constant factor 1.2.
5. Now change the data type of the matrix `m` to `np.int`. Note: Use the method `astype` of NumPy-Array.
6. Multiply all entries of the matrix `m` again with the constant factor 1.2. What data type is the result now?
7. Calculate the Hadamard product[2] (element-wise multiplication of the matrix `m` with itself (`m`⊙`m`)).

Exercise 3 — Basic operations with images - 0 Points - Programming task
This task intends to help you to set up the first contact with images in Python. To do this, you should manipulate given images and record all steps in a Python script. While implementing this task you are allowed to use only NumPy and Matplotlib, as well as `skimage.io.imread` and `skimage.io.imsave` to load and save images.

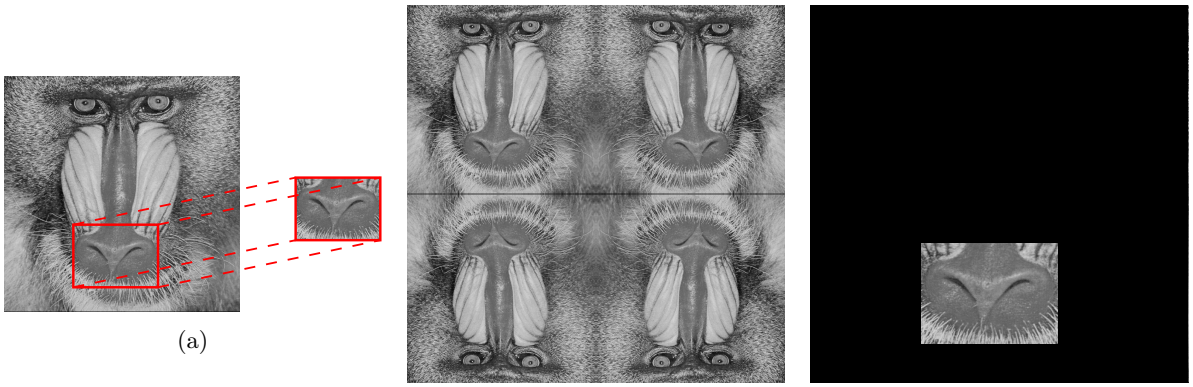
1. Save the image `mandrill.png` from Moodle and load it in Python using the function `skimage.io.imread`.
2. Show the image using Matplotlib.
3. Create a new array as an image, which visualization is the same as in Figure 1a, depicting the tip of the nose of the mandrill.
4. Save the image using the function `skimage.io.imsave` .

*This is a translation of Exercise Sheet 1 originally prepared by Christian Wilms and Simone Frintrop for the course Bildverarbeitung SS2020

[1] https://cs231n.github.io/python-numpy-tutorial/

[2] https://de.wikipedia.org/wiki/Hadamard-Produkt

5. Create a copy of the image and set a value of a single pixel (array element or image element) to black. Display the manipulated image. Can you tell the difference? (no answer text required)
6. Make another copy of the image and set the entire area of the mandrill's eyes to black. Display the manipulated image again. Can you see the difference now? (no answer text required)



(a) (b) (c)

Abbildung 1: **(a)**: Crop of the tip of the nose from the original image (left) as a new image. **(b)**: Composition of four, partially mirrored original images. **(c)**: Masked original image in which only the tip of the nose is presented.

Exercise 4 — Advanced image processing - 0 Points - Programming task

In this task, further manipulations are to be tested on an example image. To do this, write a Python script that contains all the steps for your manipulations. While implementing this task, you are allowed to use only NumPy, Matplotlib and `skimage.io.imread` to load images.

1. Create a new image based on three variations of the image `mandrill.png` from the Moodle as well as the original image.
   a) First, load again the image `mandrill.png` from Moodle in Python.
   b) Reflect the image on the vertical axis (left and right side switched) and save the result in a new variable as the first variation.
   c) Reflect the original image (`mandrill.png`) now on the horizontal axis and save the result again in a new variable. This is the second variation.
   d) Perform now both reflections one after another on the original image, save the result again in a new image as the third variation.
   e) Now create a new image that is twice as high and wide as the image `mandrill.png`. Put one of the four versions of the image in each quadrant of the new image. The original image should be visible at the top left, the image mirrored on the vertical axis on the top right, the image mirrored on the horizontal axis at the bottom left and the image mirrored on both axes at the bottom right. The result should look something like in Figure 1b. Hint: In addition to `np.array`, NumPy has several further functions for creating arrays or images.
2. Now create a negative of the original image (`mandrill.png`). In a negative, all brightness values are reversed. I.e. black becomes white, white becomes black and all values in between are reversed in the same way. Check the result by visualizing the reverse image.
3. Crop the tip of the nose of the mandrill as shown in Figure 1a as a new image. Now change a pixel in the newly created crop and find out whether this change is also happened in the original image.
4. Now create a mask for the tip of the nose of the mandrill.
   a) To do this, first create a new image that consists only of zeros and is the same size as the original image. This image becomes your mask.
   b) Now set all pixels in the mask that are in the area that contains the tip of the nose to 1.
   c) Then combine the mask and the original image so that the area outside the tip of the nose is black and the actual image can only be seen in the area of the tip of the nose. The result should look something like the figure 1c.

Exercise 5 — Broadcasting vs. Loops in NumPy - 0 Points - Programming task
Now let's measure the advantages of broadcasting in NumPy over loops. For this purpose, a query on the image should be applied once with loops and once with broadcasting. In addition, the execution time is measured using the function `time.time`. An example, how to use the function `time.time` to measure the execution times, can be found below. While implementing this task, you are allowed to use only NumPy and `skimage.io.imread` to load images as well as function `time.time` to measure the time.

1. Write a Python function that accepts an image as a NumPy array and checks for each array entry or pixel whether the pixel value is between 99 and 200. This should be implemented by iterating over the image using two `for`-loops.
2. Write a second function for the same purpose. This time with broadcasting, i.e. completely without the loops.
3. Measure how long does it take for each function to make 100 calls on the image `mandrill.png` from the Moodle. You can apply loop to call the two functions. How big is the difference?

```
>>> import time
>>> tic = time.time()
>>> #your magic here
>>> toc = time.time()
>>> diff = toc-tic
>>> print(diff)
```

Exercise 6 — Image statistics in NumPy - 0 Points - Programming task
This task is about finding, combining and efficiently using NumPy functions on images. While implementing this task, you are allowed to use only NumPy functions and `skimage.io.imread` to load images. The use of loops to iterate over images is explicitly not allowed.

1. First load the image `mandrill.png` in Python.
2. Determine the minimum, maximum, average and standard deviation of the image using the corresponding NumPy-functions.
3. Locate the minimum and maximum in the image. To do this, output the x and y coordinates of a minimum and maximum on the console. Hint: Take a look at the documentation for the corresponding functions to find out how you can generate two coordinates from the one result index.
4. Count how many pixels there are with an even or an odd value in the image. Again, don't use loops for this!
5. Also determine all coordinates at which pixels with an even value are located. The way of the output of the coordinates is irrelevant, for example: $((x_1, y_1), (x_2, y_2), (x_3, y_3), \dots)$ or $((x_1, x_2, x_3, \dots), (y_1, y_2, y_3, \dots))$.