General Course Info

Basic Image Processing
Fall 2020

Administration

NEPTUN: all of the students in the same laboratory course (code 01)

Moodle: https://moodle.ppke.hu → [P-ITJEL-0014] Basic Image Processing Algorithms
all of the students see the same materials regarding the labs

Teams: Basic Image Processing Algorithms (P-ITJEL-0014) "00"
the students will be divided into 3 channels:

channel **A**:	**IPCV** students; **Erasmus** students practice leader: Márton Bese Naszlady
channel **B**:	**BSc** students with Family name **from A* - to Stelcz** practice leader: Örkény H. Zováthi
channel **C**:	**MSc** students; **BSc** students with Family name **from Szabó - to Zs*** practice leader: Miklós Koller

Rules

- On each lab some programming tasks will be given to you. ***You have to work on the tasks during lab-time and follow the instructions of the practice leader*** in order to solve the exercises. ***You have to understand the code and be able to explain it.***
- The lab exercises will be written in MATLAB. Your uploaded Lab-solutions and Midterm-files should be able to run in version R2018b or higher without error.
- ***Using someone else's code without citation is strictly forbidden!*** Code plagiarism instantly leads to denial of the Teacher's signature.

Requirements

- ***The exercises of the Labs have to be solved and submitted within one week.*** The deadlines are clearly indicated in the Moodle system.
  The deadlines are always Mondays 23:59:59.
- There will be ***two Midterm tests in lab-practice time*** to test your knowledge regarding the lab-practices (*these are independent from the theory midterms*). The planned dates are **6th October** and **24th November**.
- The ***retake-Midterm*** (either if you have not reached 40% on any midterm or if you justifiably missed it) will be held on **7th December** (jointly with the theory-retake).

There will be a total of 10 graded Labs and 2 Midterms. In order to get the Teacher's signature the Student must

- collect at least 6 points from the Lab submissions (maximum 10), AND
- at least 40% must be achieved in both Midterms individually.

Grading

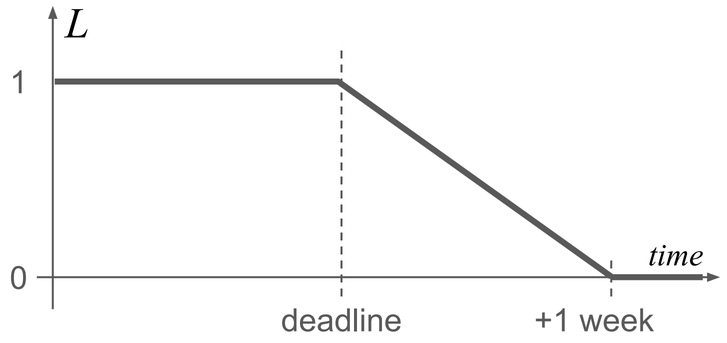
For each Lab, the grade of a submission is from the set $\{-1\} \cup [0, 1]$.

The grade of a Lab is -1 if the submission is missing or the code throws a serious error (e.g. `MATLAB:UndefinedFunction`, `MATLAB:badsubscript`, `MATLAB:TooManyInputs` etc.)

Otherwise, the Lab's grade is a number from the $[0,1]$ interval depending on how well the required functions were implemented.

In case of a late submission the $[0,1]$ range is scaled down by a factor L which depends on the date of submission.

Each task can be submitted only once!



Lab 1

Basic Image Processing
Fall 2020

Data in MATLAB

- how to create a matrix:
  - `m1 = [22 12 42 32; 62 52 72 92];`
  - `m2 = [11; 33];`
- transpose: `m2t = m2';`
- concatenate: `m3 = [m1, m2];`
- indexing:
  - **index itself starts from 1!**
  - single element: `element = m1(2, 1)`

22	12	42	32
62	52	72	92

11
33

11	33

22	12	42	32	11
62	52	72	92	33

62

```
m1(1, end) = 77
```

special keyword

22	12	42	77
62	52	72	92

Data in MATLAB

`m3:`

22	12	42	32	11
62	52	72	92	33

- indexing:
  - linear indexing:
    - in the background, your ND array stored as a 1D column vector
    - with one index only (regardless the number of dimensions) you can refer to any of your element

      `element = m3(3)` → 12

      `m3(5) = 113` →

22	12	113	32	11
62	52	72	92	33

  - accessing multiple elements aka. *indexing with vectors/matrices*:
    - consecutively: `mp1 = m3(2, 2:5 )`

2	3	4	5

→

52	72	92	33

    - non-consecutively: `mp2 = m3(1, 1:2:5 )`

1	3	5

→

22	113	11

Data in MATLAB

`m3:`

22	12	113	32	11
62	52	72	92	33

- *indexing / acc. mult. elements continued:*
  - along a whole dimension/direction: `m3(:, 3) = 19`

22	12	19	32	11
62	52	19	92	33

  - with logical array / expression:

```
idxs = logical([1 0 0 0 1; 1 0 0 0 0])
m3(idxs)
```

1	0	0	0	1
1	0	0	0	0

22
62
11

```
idxs2 = m3>20 & m3<40
m3(idxs2)
```

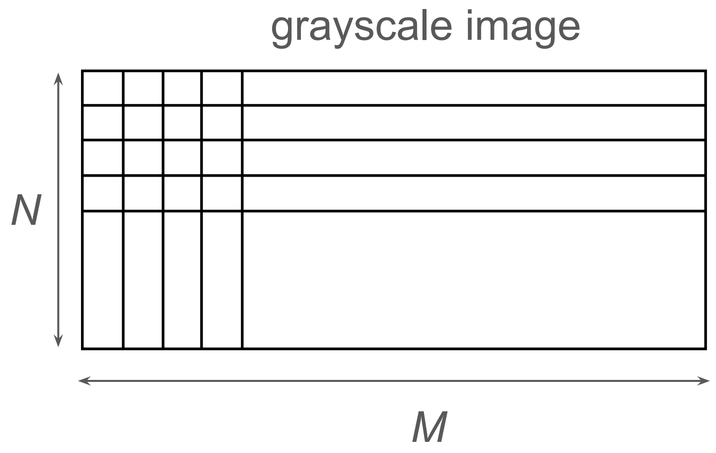
1	0	0	1	0
0	0	0	0	1

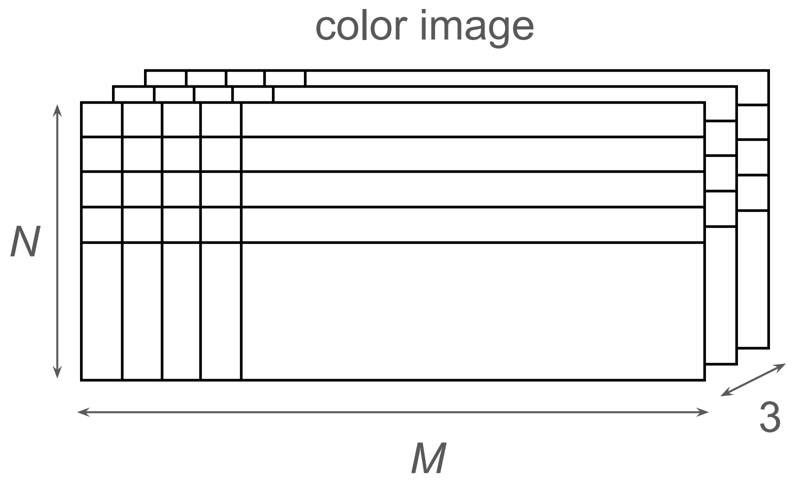
22
32
33

- `size, squeeze, reshape` (please recap from MATLAB Help these & the chapter Matrix Indexing)

Image data in MATLAB

- **Picture elements (pixels) have a location** (vertical, horizontal, (and channel) coordinates) **and an intensity value.**
- role of the type of the data
  - `uint8` → range: [0, 255]
  - `double` → range: [0, 1]
    `double` is the default type of every number in MATLAB
- number of dimensions of the array
  - 2 → grayscale or BW image
  - 3 → color image (channels ‘r’, ‘g’, ‘b’)





Script vs. Function

Both:

- created in the Editor, saved as *.m file
- sequence of commands, as typed in the Command Window

● **no formal constraint** on the syntax
● the **workspace is shared** with other, individual commands typed in the Command Window
	● **syntax constraints**:
	first line:
	`[out_args] = function fname(in_args)`
	last line:
	`end`
	the `fname` should match the name of the m-file
	● the **workspace is individual**:
	○ when terminates, all the variables are destroyed except return values
	○ during execution, the outer workspace is unreachable

You will have to hand in functions only!

For every Lab task you can download a “code package” containing all the necessary test images, scripts and empty functions.

Your task is to implement the functions as described in the “slides” of the Lab.

After implementation, you should run the scripts, understand the whole project.

Finally, you have to upload the implemented function files only!
(I.e. no test images, scripts, results, reports etc. are needed).

But today is somewhat special :P

Now please

download the ‘Lab 01’ code package

from the

moodle system

Exercise 1

Create a script (`script1.m`) in which:

- Using `imread()`, load the file ‘AlfredoBorba_TuscanLandscape.jpg’
- Using `imshow()`, display the loaded image
- Create a logical variable which tells whether the image is color
  - Name the variable `isColor`
  - Check if the number of channels is exactly 3. For this you can use `size()`.
- Using `rgb2gray()`, convert the image to grayscale
- Using `imshow()`, display the grayscale image
- Using `imwrite()`, save the grayscale image into the output folder. Name it ‘AlfredoBorba_TuscanLandscape_GRAY.jpg’

Run this script and examine the results.

Exercise 2a

Create a script (`script2.m`) in which:

- Using `imread()`, load the file ‘AlfredoBorba_TuscanLandscape.jpg’
- Call the function `flip_and_rotate()`
  - Check the empty function file to know the input and output arguments
  - In Exercise 2b you have to implement this function
- Using `subplot()` and `imshow()`, display the three returned images side by side.

After implementing the function, run this script and examine the results.

Exercise 2b

Implement the function `flip_and_rotate()` in which:

- The first returned value `VER` have to be the vertically flipped input image. Use `flipud()`.
- The second returned value `HOR` have to be the horizontally flipped input image. Use `fliplr()`.
- The third returned value `ROT` have to be the input image rotated by 45 degrees clockwise. Use `imrotate()`.

Exercise 3

Implement the function `swap_RB_dumb()` in which:

- In the returned image `BGR` the red and blue channels of the input image have to be swapped.
- **You have to carry out the operation pixel by pixel,** using a `for` loop.

Implement the function `swap_RB_smart()` in which:

- In the returned image `BGR` the red and blue channels of the input image have to be swapped.
- **You have to carry out the operation using array indexing,** without any loops.

Run the script `script3.m` to test your functions, measure the runtimes and visualize the result. Compare the runtimes. *Remember, that doing matrix operations using loops is usually slower than using smart indexing!*

Dumb: 0.011179 seconds

Smart: 0.000878 seconds

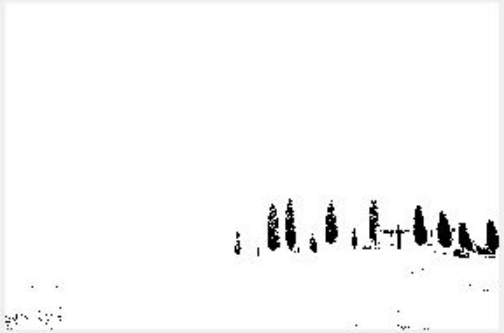
Exercise 4

Implement the function `threshold()` in which:

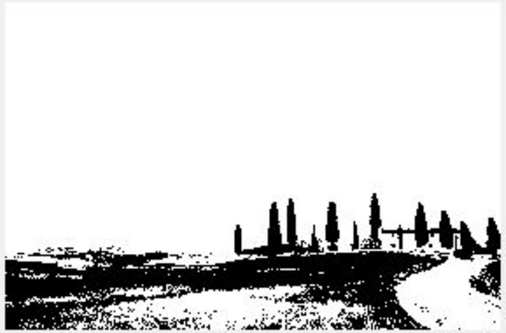
- Check if the input image `IMG` is grayscale (has only one channel). If not:
  - display a warning (use `warning()`),and
  - convert the input to grayscale (use `rgb2gray()`).
- The returned value `TH` should be a black-and-white image {0, 255} where all the pixels below the parameter `level` are black, the others are white.

Run the script `script4.m` to test your function and visualize the results of different threshold levels.

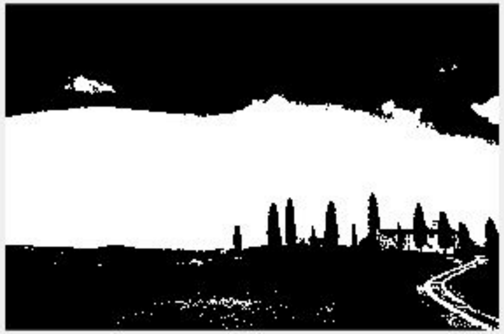
level = 50



level = 100



level = 150

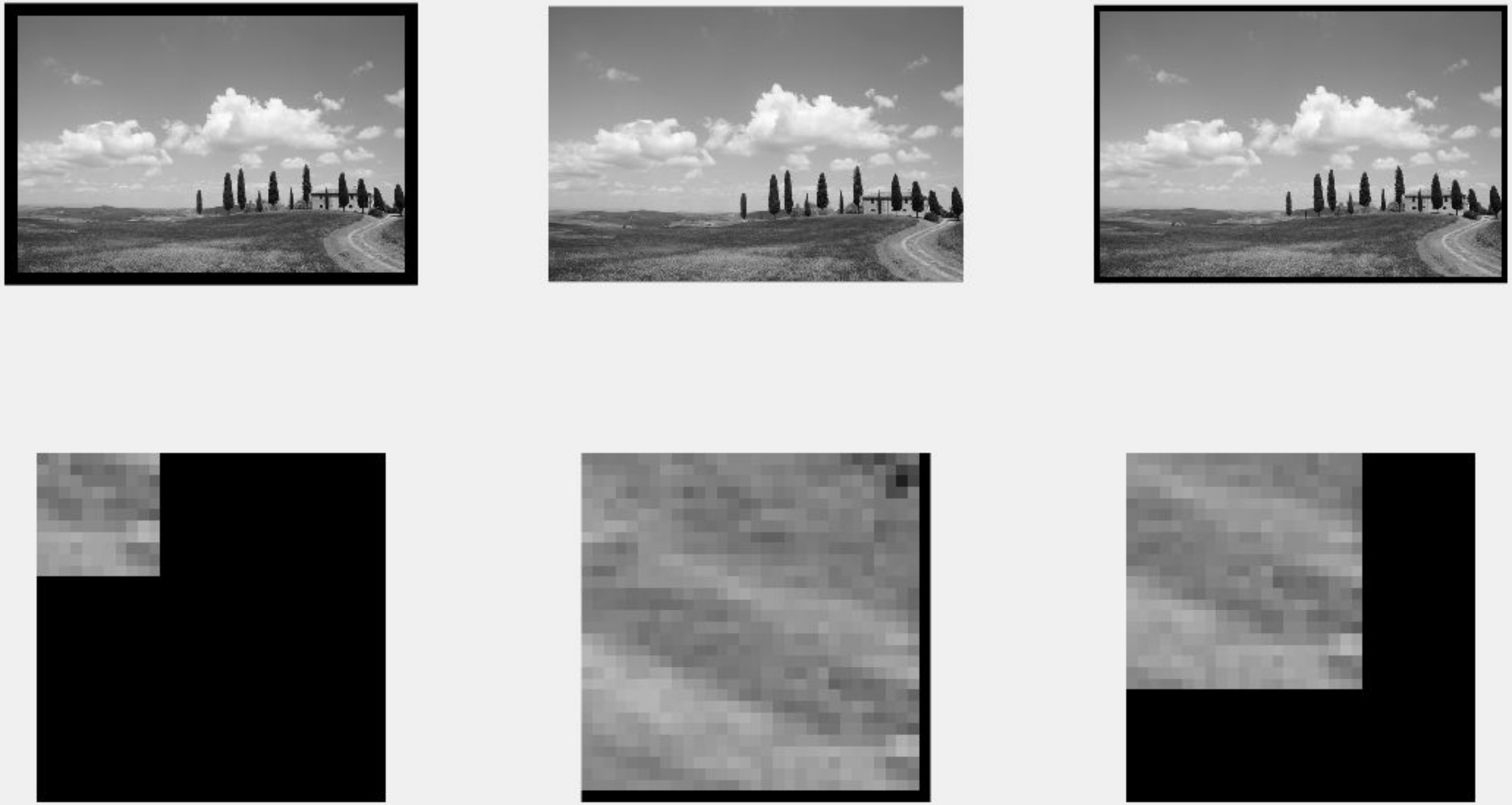


Exercise 5

Implement the function `pad_image()` in which:

- The function should work with 1 or 2 input arguments.
  - Use `varargin`.
  - The first input is always the input image `IMG`.
  - If given, the second one is the border size (`border_size`).
  - If there is no second argument, the border size should be 10.
- Check if the input image `IMG` is grayscale (has only one channel). If not:
  - display a warning (use `warning()`),and
  - convert the input to grayscale (use `rgb2gray()`).
- The returned value `PAD` should be the grayscale image surrounded by zeros.
  - It's advised to create a bigger matrix filled with zeros and copy `IMG` in the center.

Run the script `script5.m` to test your function and visualize the results.

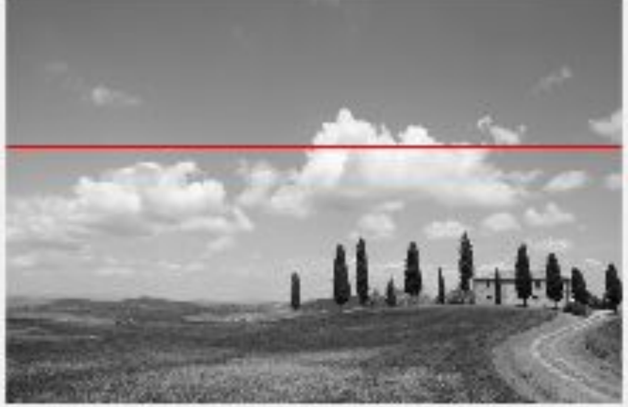


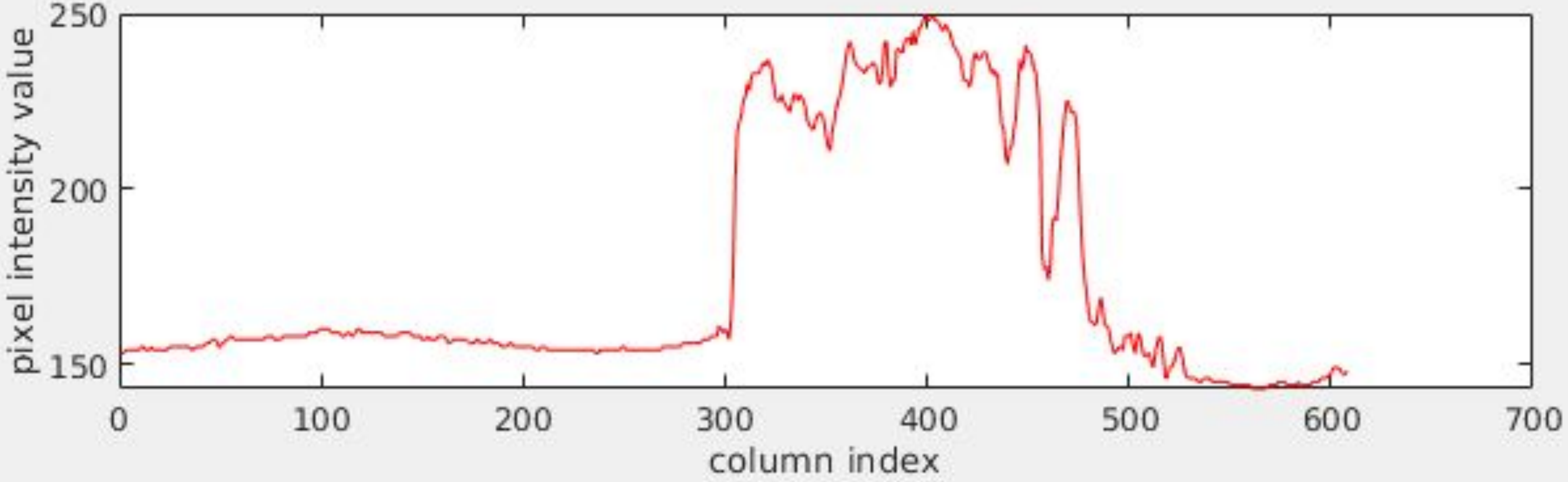
Exercise 6

Create a script `script6.m` in which:

- You should load the file 'AlfredoBorba_TuscanLandscape.jpg' and convert it to grayscale.
- Define a variable (`row`) and set its value to 150.
- Raise a new figure, display two subplots:
  - In the top, show the grayscale image.
  - On the grayscale image, plot a line according to the row number.
  - In the bottom, plot the pixel intensity levels in the selected row as a function of the column index.
  - Set the axis labels as seen in the next slide.

Run the script, visualize the result, play with the value of `row`.




250
200
150
pixel intensity value
0
100
200
300
400
500
600
700
column index

THE END