Mini-project in Image Processing

Purpose:

To ensure that each student gets experience with IP (through Python/OpenCV) by implementing an algorithm from scratch.

Form:

The mini-project is an individual self-study. No dedicated lectures.

Assistance:

Peer learning, but you are also welcome to contact us (Anders, Eoin, and Tsampikos).

Hand-in:

Each student makes a document that contains 1) an explanation of their algorithm, 2) their code, 3) an explanation of their code and 4) a documentation of the program (show input and output) and showing the effect of different parameters on the algorithm, if any. Everything is collected in ONE PDF file per student. The document should be named like this: ”your topic_your name#”.pdf
Caution : Failing to follow the instructions means failing the mini-project.

Presentation:

Present your work for your group members (in your group rooms) on 19th of November at 9:30- 12:00. We will drop by and listen to some of the presentations.

Deadline for handin:

Upload to Moodle no later than **26th of November at 16:00**.

Exam:

Handing in on time AND having your work approved is a prerequisite for participating in the image processing exam in January.

Recommend procedure:

- Find literature about your topic (see your IP book, other books, papers)
- Design your algorithm on paper
- If you are not sure about the correct algorithm ask Tsampikos.
- Implement the algorithm in Python (using the allowed OpenCV functions)

- Document the work (in ONE PDF file) and upload to Moodle.
- Prepare a short presentation for your group.

Allowed OpenCV functions:

- **Imread**
- **Imshow**
- **waitKey**
- **All matplotlib functions**

Topics for the mini-projects:

The following projects are distributed randomly among the members in each group. Students in the same group will all work on different topics. Each project must as a minimum contain the following elements: variables, a function (main() does not count), a loop, and an if-else statement. It is not allowed to use library functions when implementing your algorithm, for example to rotate an image, but it is fine to use OpenCV functions for loading/displaying images.

Topic #1: Rotate

Make a Python program that can rotate an image q degrees around the point (x,y) using both forward and backward mapping.
Input: Grayscale image, q, x, y
Output: Visualization of the rotated image

Discussion: Compare the results of forward and backward mapping.

Topic #2: Shearing

Make a Python program that can shear an image in the x-direction with a factor of Bx and in the y-direction with a factor of By, using both forward and backward mapping.
Input: Grayscale image, Bx, By
Output: Visualization of the sheared image

Discussion: Compare the results of forward and backward mapping.

Topic #3: Scaling

Make a Python program that can scale an image in the x-direction with a factor of Sx and in the y-direction with a factor of Sy.
Input: Greyscale image, Sx, Sy
Output: Visualization of the scaled image

Discussion: Compare the results of forward and backward mapping.

Topic #4: Histogram equalization

Make a Python program that can improve the contrast in an image using histogram equalization.
Input: Grayscale image
Output: Visualization of the transformed image

Topic #5: Diagonal Edge Detection

Make a Python program that can find diagonal edges in an image.
Input: Greyscale image
Output: Binary image where the diagonal edges are white (255) and the rest of the pixels black (0)

Topic #6: Convert from RGB to HSI

Make a Python program that can convert from RGB to HSI. Input: An RGB image Output: Three
images: a H-image, a S-image, and an I-image

Topic #7: Horizontal and Vertical Edge Detection

Make a Python program that can find horizontal and vertical edges in an image.
Input: Greyscale image
Output: Binary image where the diagonal edges are white (255) and the rest of the pixels black (0)