Computer Vision course

Proff. Stefano Ghidoni | Matteo Terreran

Lab 1 - Intro to OpenCV, pixel manipulation, transformations

Note: try compiling at least one task from the command line.

Task 1
Implement the OpenCV Hello world discussed during the first lectures, adding:

- A safety check on `argc` - check that the image is provided as a command-line argument. If `argc < 2`, print a message warning the user that an image filename shall be provided;
- a safety check on the image returned by `cv::imread()` - what should happen if the filename is wrong? Check on the `cv::imread()` documentation and handle such a condition properly.

Use the executable to open the grayscale and color images provided and test the safety checks above.

Task 2
In a new file, edit the software in Task 1 by adding instructions for printing the number of channels of the image. Also, write the output of `cv::waitKey()` into a `char` variable and print it before exiting. Check the OpenCV documentation (or the slides) if you do not know how to get the number of channels. Run the executable on all the images provided.

Task 3
In a new file, edit the software in Task 2 by adding a function that checks if the number of channels of the input image is 3. If so, it sets to 0 the first channel and visualizes the image. Which color is missing? Try other versions that set to 0 the second or the third channel. Which color is missing? What is the color coding used by OpenCV? Try with all the images provided.

Task 4
Edit the software in Task 2 by adding a function that checks if the number of channels of the input image is 3. If so, create three images with the same size of the input image, one channel, containing the values found in the first, second and third channel of the original image. Visualize such images. Try with all the images provided.

Task 5

Write a program that creates two images of size 256x256, one channel, 8-bit depth unsigned char. Write the pixels using the `Mat::at()` function in order to create:

- a vertical gradient in the first image;
- a horizontal gradient in the second image.

Show the images on screen. Expand the software to create and visualize two other images, size 300x300, same features as above, with:

- a chessboard with squares of size 20 pixels;
- a chessboard with squares of size 50 pixels.

Task 6

Write a program that opens an image whose filename is provided via command line argument, converts it to grayscale using the `cv::cvtColor()` function and calculates, from the grayscale image:

- the output of an averaging filter;
- the output of a Sobel filter;
- the output of a max filter;
- the output of a min filter.

Feel free to choose the size of the kernel and to choose an implementation of the averaging and Sobel filters. The program then shows the input and output images on screen.