Lab 1 - Introduction to OpenCV

- IDE configuration and first OpenCV examples.
- Image operations; reading and displaying images with different formats, direct pixel manipulation.
- Example of a mathematical operation: image subtraction.
- Interaction: selecting pixels and drawing on an image.
- Conversion between color spaces.

1.1 OpenCV Installation and Documention

Installation

Follow the steps in the following link to perform Python and OpenCV installation for python:

https://www.python.org/downloads/

https://pypi.org/project/opencv-python/

Documentation

The OpenCV documentation is available at:

http://docs.opencv.org

1.2 First example

Run and test the file **Aula_01_ex_01.py**

Analyze the code and the OpenCV functions that are used.

Note how an image is read from file and displayed.

1.3 Direct pixel manipulation

Read again the lena.jpg image and create a copy of the image (function copy).

Access the pixels values of the copied image, set to 0 every pixel of the copy image whose intensity value is less than 128 in the original image, you can access a given pixel as an array image[x,y].

Display the original image and the modified image.

Modify the code to allow reading the name of the gray-level image from the command line. Do not forget to import the system library (import sys) to allow the access to the command line arguments (sys.argv[1])

1.4 Simple mathematical operation: image subtraction

Based on the previous example, create a new program that reads and displays the two image files **deti.bmp** and **deti.jpg**.

To identify possible differences between the two images, carry out a **subtraction** operation. Be careful since the cv subtract operation is saturated and is different from the numpy – operations that is a modulo operations.

Analyze the resulting image.

Optional

Open an image of your choice in an image editor and save it on file using the **jpeg** format with different compression ratios.

Compare the results of the image subtraction operation for different compression ratios.

1.5 Interaction: selecting a pixel and drawing a circle

Modify the previous example to open and display just one image. Add a callback function to detect a right mouse click on the window and draw filled circle should be drawn, with center on the selected image pixel (function cv2.circle).

To register the new callback function use:

```
def mouse_handler(event, x, y, flags, params):
  if event == cv2.EVENT_LBUTTONDOWN:
      print("left click")
```

Do not forget to associate the callback to each window using the following code:

```
cv2.setMouseCallback("Window", mouse_handler)
```

1.6 Conversion between color spaces

Load a color image and use the function **cvtColor** to convert it to a gray-level image (COLOR_RGB2GRAY).

Optional

Consult the documentation for the function **cvtColor** and modify the example to visualize the image in different color spaces (for instance: COLOR_RGB2HLS, COLOR_RGB2XYZ, COLOR_RGB2HSV)