Lab 1

- 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.

Documentation

The OpenCV documentation is available at: http://docs.opencv.org

OpenCV on Linux/UBUNTU

The installation process is detailed, for instance, at:

http://www.codebind.com/linux-tutorials/install-opencv-ubuntu-18-04-lts-c-cpp-linux/

Alternative – Visual Studio configuration

The configuration of an OpenCV project in Visual Studio, using C++, is detailed, for instance, at:

http://opencv-srf.blogspot.pt/2017/11/install-opencv-with-visual-studio.html

To use the OpenCV library in a C++ project, the following steps are required (**adapted to your particular folder structure**):

1. Create an empty **Win Console Application** project.

2. Set the compilation for 64 bits.

   Select **Project Property Pages -> All Configurations -> x64**.

3. Indicate the folder containing the OpenCV header files and folders.

   Select **Project Property Pages -> All Configurations -> Configuration Properties -> C/C++ -> General**.

   In the **Additional Include Directories** field indicate the folder containing the header files and folders.

4. Indicate the folder containing the various OpenCV library files.

   Select **Project Property Pages -> All Configurations -> Configuration Properties -> C/C++ -> General**.

   In the **Additional Library Directories** field indicate the folder containing the library files.

5. Update the system PATH variable.

   Select **Project Property Pages -> All Configurations -> Configuration Properties -> C/C++ -> Debugging**.

   In the **Environment** field update the system variable, similarly to PATH=c:\opencv331\build\x64\vc14\bin;%PATH%

6. In the **Configuration Manager** − both for the **Debug** configuration and for the **Release** configuration − set the compilation for 64 bits (**x64**).

7. Associate library file **opencv_world331d.lib** to the **Debug** configuration and the library file **opencv_world331.lib** to the **Release** configuration.

   Select **Project Property Pages -> Debug -> Linker -> Input -> Additional Dependencies -> Edit** and add **opencv_world331d.lib**.

   Select **Project Property Pages -> Release -> Linker -> Input -> Additional Dependencies -> Edit** and add **opencv_world331.lib**.

1.1 First example

Compile and test the file **OpenCV_ex_01.cpp**

Analyze the code and the OpenCV functions that are used.

Note how an image object is instantiated, and an image is read from file and displayed.

Task

Consult the attributes of the **image** object to get its size, number of channels and number of bytes per pixel, and write them on the console window.

1.2 Direct pixel manipulation

Tasks

Create a copy of the image using the method **cv::clone**.

Using the **data** attribute of the copy image, set to 0 every pixel of the copy image whose intensity value is less than 128 in the original image.

A better alternative to get or set image pixel values is to use the generic template method **at** which allows using row and column indices:

```
image.at<uchar>(y,x)
```

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.

1.3 Simple mathematical operation: image subtraction

Tasks

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.

Analyze the resulting image.

Image subtraction can be performed either using the subtraction operator (**–**) or calling the function **subtract**.

(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.

1.4 Interaction: selecting a pixel and drawing a circle

Task

Add to the previous example a ***callback function*** to detect a ***mouse click*** on a displayed image.

To register the new callback function use:

```
        setMouseCallback( const string& winname,
                          MouseCallback on_mouse, void userdata=0 );
```

The callback function has the following prototype:

```
  void on_mouse( int event, int x, int y, int flags, void *param );
```

When pressing the right mouse button, a filled circle should be drawn, with center on the selected image pixel.

1.5 Conversion between color spaces

Task

Load a color image and use the function **cvtColor** to convert it to a gray-level image (CV_RGB2GRAY).

(optional)

Consult the documentation for the function **cvtColor** and modify the example to visualize the image in different color spaces (for instance: CV_RGB2HLS, CV_RGB2XYZ, CV_RGB2HSV).