Lab session

Introduction to OPENCV

Fall 2018 course: *ImProc*
Digital Image Processing

October 23th 2018
15h – 16h45

J.-L. Dugelay, Badr Tajini, Chiara Galdi, Valeria Chiesa, Khawla Mallat

EURECOM
Security department

Send your final report by October 30th 2018
e-mail: {To: tajini, CC:jld}@eurecom.fr

EURECOM
Sophia Antipolis

I. General advices

I.A. General warning in Visual Studio

- If you see the following warnings: ''This project is out of date''. Click on ''do not show this dialog again''.
- (Mandatory step) Go to Tools->Options->Debugging->Symbols and select check-box ''Microsoft symbol servers''.

I.B. Path of executable file

If you want to use the command window, you must use the command line *pushed* to change the directory. The path to run the program in command window (i.e. the path where your executable file is) is:

WINDOWS PATH	**E:\TPData\Tp_OpenCV\Debug**

Copy the folder from ***\\datas\teaching\courses\image\TP_OpenCV*** to ***E:\TPData***

I.C. Description of the problem

The goal of this Lab Session is to get familiar with OPENCV. In this lab, you will do the following

1. Compile the example code in “ExampleCode.cpp” which has already been filled to see the output and some example function calls in OPENCV (Section II);

2. Fill the missing parts of the code in “imgprocessing.cpp” to obtain the expected outputs (Section III).

You will open the existing project **E:\TPData\Tp_OpenCV\TPOpenCV.sln** using Visual Studio 2017. The paths for OPENCV have already been integrated to the project.

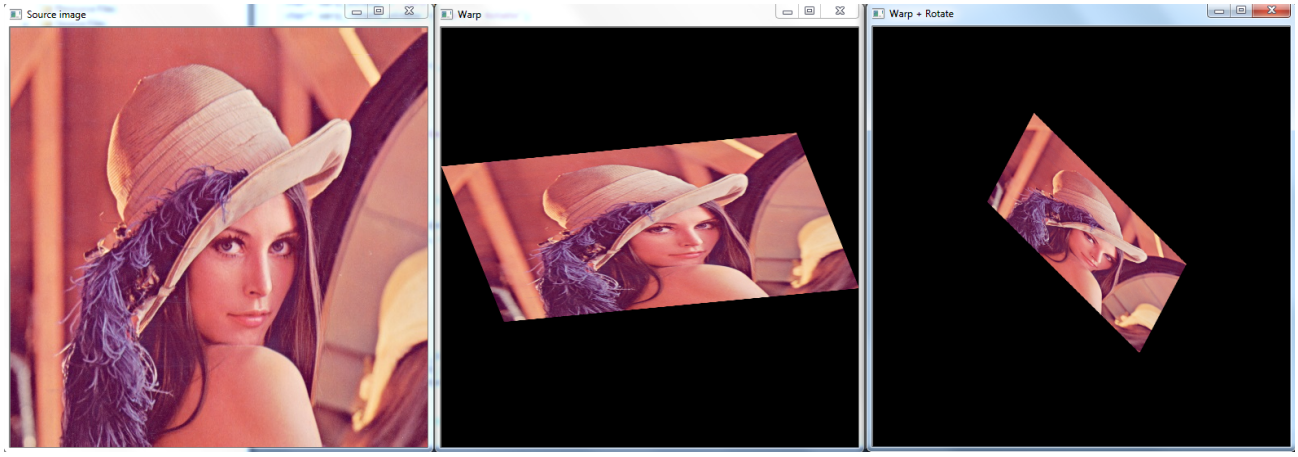
To interchange the code file (.cpp) from Solution Explorer, right click and exclude the code file from the project. Next, right click on Source Files under Solution Explorer and add an existing item “.cpp” under **E:\TPData\Tp_OpenCV\TpOpenCV**.

II. Example Code for Warping and Rotation

In Section II, you will use the image named *Lenna.jpg*; it is already present in the path **E:\TPData\Tp_OpenCV\imgs**. First clean, rebuild and build the solution, then, open a command window and go to the directory of your path. Then in the command window (cmd), type “TpOpenCV ../imgs/Lenna.jpg”.

You can also rebuild the solution from Visual Studio. Go to Project → TpOpenCV Properties → Configuration Properties → Debugging → Command arguments and add the link of your image that you want to use.

The expected output images of this section are as follows:



III. Image Processing Implementations in OPENCV

Note: The expected output images of this exercise are shown at the end of this document.

III.A. Step 1: load the image and convert it to a grayscale image

After compiling the example code in Section II, in this section, you now have to fill the missing parts in the file "imgprocessing.cpp". For this purpose, first select ExampleCode.cpp from Solution Explorer, and right click and exclude the file from the project in case you did not exclude it in Section II.

Next, right click on Source Files under Solution Explorer and add an existing item "imgprocessing.cpp" under **E:\TPData\Tp_OpenCV\TpOpenCV**.

In this step, you will load the image and assign it to the Mat variable "src", then convert it to a grayscale image. In Section III, you can benefit from the information included in the link "http://docs.opencv.org/doc/tutorials/imgproc/table_of_content_imgproc/table_of_content_imgproc.html"

When compiling this program, similar to the example code introduced in Section II, you will open a command window and go the directory of your path (**E:\TPData\Tp_OpenCV\Debug**). Then in the command window, type "TpOpenCV ../imgs/Baboon.jpg".

You can also rebuild the solution from Visual Studio. Go to Project → TpOpenCV Properties → Configuration Properties → Debugging → Command arguments and add the link of your image that you want to use.

In this step, you might use the following functions: , *imread, imshow, cvtColor*

III.B. Step 2: apply Histogram Equalization

In this step, you will apply histogram equalization to the grayscale image (Output of Step 1).

You might use the following functions: *equalizeHist*

III.C. Step 3: apply remapping; first turn the image upside down and then reflect the image in the x direction

In this step, you will apply upside down operation and reflection in the x direction to the image for which histogram equalization is applied (Output of Step 2) and show the images computed. You can check the Remapping section in the OpenCV tutorial to fill the missing parts in the code.

You might use the following functions: *remap*

III.D. Step 4: apply median filtering

In this step, first, you will apply median flitering to the image for which histogram equalization is applied (Output of Step 2) and show the image computed. You can check the Smoothing section in the OpenCV tutorial to fill the missing parts in the code. The parameter for maximum kernel length is already set to 6 as a global variable for this exercise.

You might use the following functions: *medianBlur*

III.E. Step 5: apply Gaussian Filter

In this step, first, you will apply Gaussian filtering to the image for which histogram equalization is applied (Output of Step 2) and show the image computed. You can check the Smoothing section in the OpenCV tutorial to fill the missing parts in the code. Select the size of the filter for Gaussian blurring as Size(3, 3).

You might use the following functions: *GaussianBlur*

III.F. Step 6: apply Laplace Operator to compute Edge Image

In this step, first, you will apply Laplace operator to the Gaussian filtered image (Output of Step 5) and show the resultant edge image. You can check the Laplace Operator section in the OpenCV tutorial to fill the missing parts in the code.

You might use the following functions: *Laplacian*, *convertScaleAbs*

III.G. Step 7: apply Sobel edge detector to compute Edge Image

In this step, first, you will apply Sobel edge operator to the Gaussian filtered image (Output of Step 5) and show the resultant edge image. You can use the Sobel Derivatives section in the OpenCV tutorial to fill the missing parts in the code.

You might use the following functions: *Sobel, convertScaleAbs, addWeighted*

Expected Output Images of Section III