Lab session

Introduction to OPENCV

(with Python)

Fall 2018 course: *ImProc*
Digital Image Processing

November 6th 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

0. Prepare your environment

Before starting the lab. Open anaconda prompt (command line) and copy this line to update Open cv library:

pip install opencv-python==3.3.0.10 opencv-contrib-python==3.3.0.10

Then, run python in the same command line by writing:

Python

Import Open cv by writing:

import cv2

Later, copy this command line:

print (cv2 .__ version__)

This step is used to check if the Open cv library is up-to-date.

The result of this command is shown as below:

3.3.0

I. Introduction to Python and OpenCV

Python

Python is a high-level programming language. It is object-oriented and a general purpose language, ideal for scripting and rapid application development in many areas on most platforms.

OpenCV

OpenCV (*Open Source Computer Vision*) is a library of programming functions mainly aimed at real-time computer vision.

To use it on python it is important to include it on the header of the code as *import cv2*.

II. Basic Operations

A. Manage images

Read a file image: In order to load an image to memory we have to use the command *img = cv2.imread('filename').* When using *imread*, after the file name, you can include a flag to specify the way an image should be read:

- cv2.IMREAD_COLOR (The flag could also be 1): Loads a color image. *img = cv2.imread('filename',1)* or *img = cv2.imread('filename',cv2.IMREAD_COLOR)*
- cv2.IMREAD_GRAYSCALE (The flag could also be 0): Loads an image in grayscale mode. *img = cv2.imread('filename',0)* or *img = cv2.imread('filename',cv2.IMREAD_GRAYSCALE)*
- cv2.IMREAD_UNCHANGED (The flag could also be -1): Loads image including alpha channel. *img = cv2.imread('filename',-1)* or *img = cv2.imread('filename',cv2.IMREAD_UNCHANGED)*

Visualize an image: To visualize an image in python with opencv the command used is *cv2.imshow('Window title', img)*.

Save an image: Using the function *cv2.imwrite('filename', img)* the argument *filename* is the name that the image will have when saved.

B. Image properties

Shape of image: To obtain a tuple with the number of columns, rows and channel (if image is in color) it is used *rows, cols, ch = img.shape*

Image size: To get the size of an image the function used is *size = img.size*

Image data type: To obtain the image data type the function used is *type = img.dtype*

Image channels: To get the image color channels you can use the function *split* as *b, g, r = cv2.split(img)* or individually as *b = img[:,:,0]*

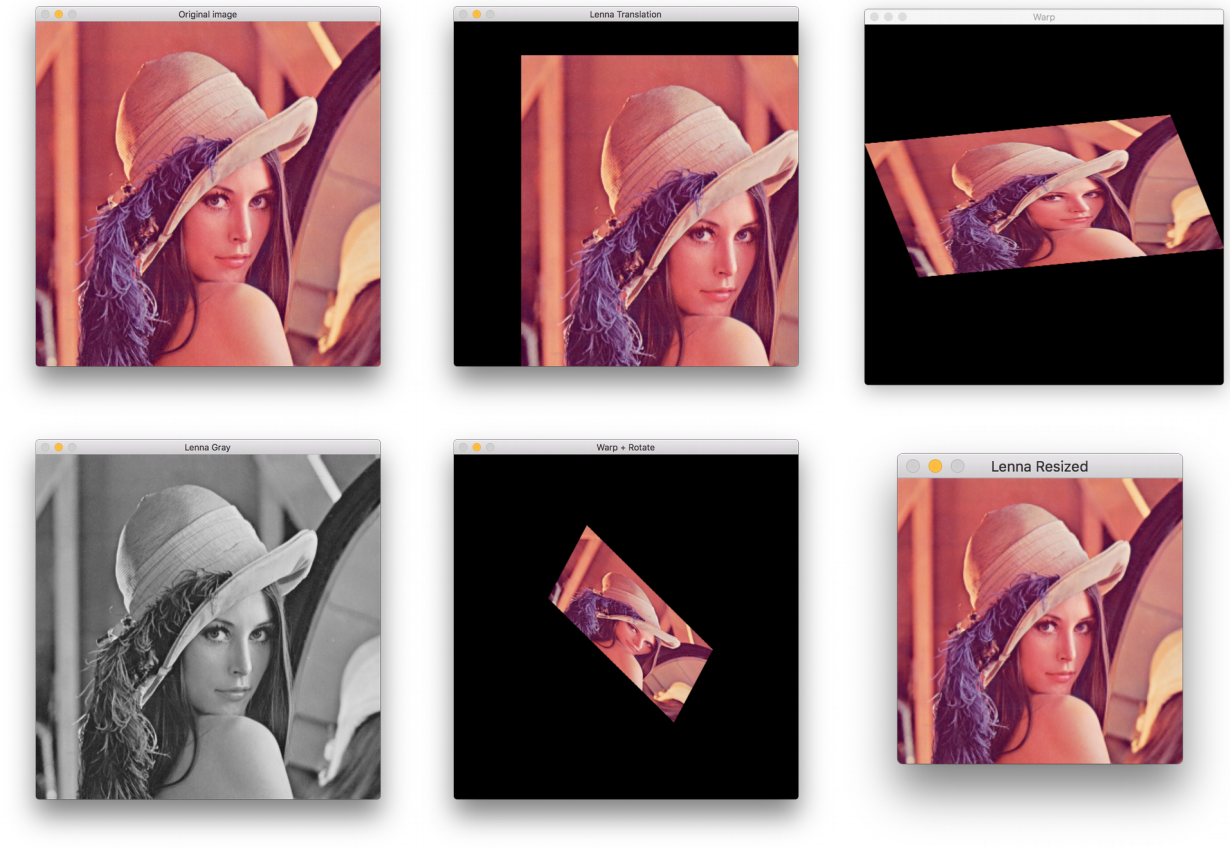
C. Example Code

The image used for the example code is *Lenna.jpg*. The example code takes the image and applies a resize, translates the image, warps and rotates it. To build the solution it is necessary to execute the code in the command window, writing ***python exampleOpenCV.py***

The functions used: cv2.resize, shape, cv2.warpAffine, cv2.getAffineTransform, cv2.getRotationMatrix2D

The output should be as follows :

III. Image Processing implementations in OpenCV



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

In this step, the image loaded will be *Baboon.jpg*.

The functions used: cv2.imread, cv2.cvtColor

B. Step 2: apply Histogram Equalization

Apply the histogram equalization to the grayscale image (Output of step 1).

The functions used: cv2.equalizeHist

C. Step 3: apply remapping; image upside down and image reflected in the x direction

Apply upside down operation and reflection in the x direction to the image to the histogram equalization image output (Output of step 2).

The functions used: cv2.remap

D. Step 4: apply median filter

Apply median filtering to the image for which the histogram equalization was applied (Output of step 2).

The functions used: cv2.medianBlur

E. Step 5: apply gaussian filter

Apply gaussian filtering to the image for which the histogram equalization was applied (Output of step 2).

The functions used: cv2.GaussianBlur

F. Step 6: apply Laplace operator to compute Edge image

Apply the Laplace operator to the gaussian filtered image (Output of step 5).

The functions used: cv2.Laplacian, cv2.convertScaleAbs

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

Apply the Soble edge operator to the gaussian filtered image (Output of step 5).

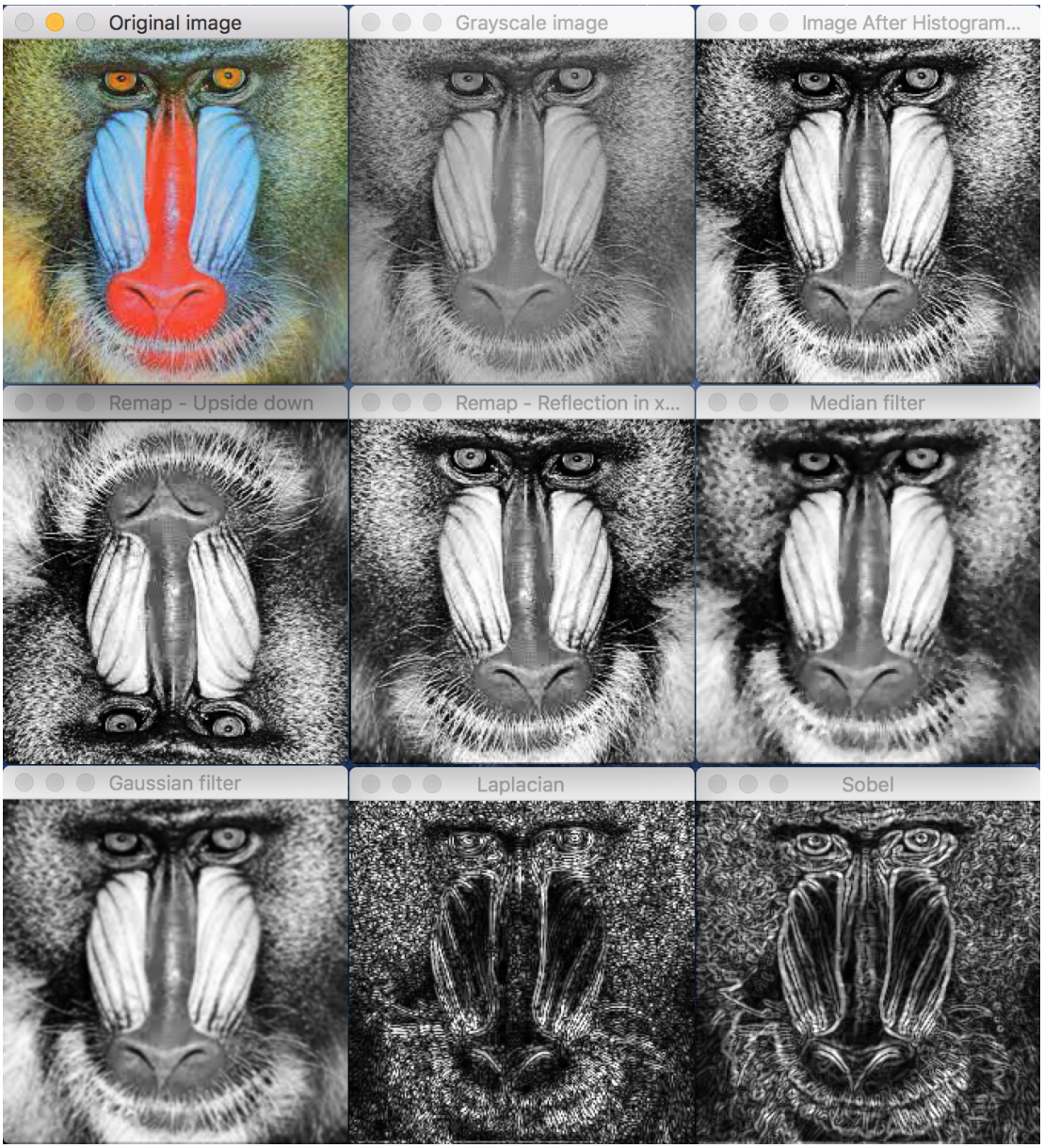
The functions used: cv2.Sobel, cv2.convertScaleAbs, cv2.addWeighted

H. Example Code

The image used for the example code is *Baboon.jpg*. To build the solution it is necessary to execute the code in the command window, writing ***python imgprocessing.py***

The functions used: cv2.imread, cv2.cvtColor, cv2.equalizeHist, cv2.remap, cv2.medianBlur, cv2.GaussianBlur, cv2.Laplacian, cv2.convertScaleAbs, cv2.Sobel, cv2.convertScaleAbs, cv2.addWeighted

The output should be as follows


Original image
Grayscale image
Image After Histogram...
Remap - Upside down
Remap - Reflection in x...
Median filter
Gaussian filter
Laplacian
Sobel