CS-512: Computer Vision- FALL 2018

ASSIGMENT 2

Diego Martin Crespo
A20432558

1. PROBLEM STATEMENT

1.1. THE IMAGE TO BE PROCESSED BY THE PROGRAM SHOULD BE READ FROM A FILE IF A FILE NAME IS SPECIFIED IN THE COMMAND LINE OR CAPTURE IT FROM THE CAMERA IF A FILENAME IS NOT SPECIFIED IN THE COMMAND LINE

1.2. THE READ IMAGE SHOULD BE READ AS A 3-CHANNEL COLOR IMAGE

1.3. THE PROGRAM SHOULD WORK FOR ANY SIZE IMAGE

1.4. SPECIAL KEYS ON THE KEYBOARD SHOULD BE USED TO MODIFY THE DISPLAY IMAGE AS FOLLOWS:

a) *'i': reload the original image.*

b) *'w': save the current image into the file 'ouput.jpg.'*

c) *'g': convert the image to grayscale using the OpenCV conversion function.*

d) *'G': convert the image to grayscale using your implementation of conversion function.*

e) *'c': cycle through the color channels of the image showing a different channel every time the key is pressed.*

f) *'s': convert the image to grayscale and smooth it using the openCV function. Use a track bar to control the amount of smoothing.*

g) *'S': convert the image to grayscale and smooth it using your function which should perform convolution with a suitable filter. Use a track bar to control the amount of smoothing.*

h) *'d': downsample the image by a factor of 2 without smoothing.*

i) *'D': downsample the image by a factor of 2 with smoothing.*

j) *'x': convert the image grayscale and perform convolution with an x derivative filter. Normalize the obtained values to the range [0,255].*

k) *'y': convert the image to grayscale and perform convolution with a y derivative filter. Normalize the obtained values to the range [0,255].*

l) *'m': show the magnitude of the gradient normalized to the range [0,255]. The gradient is computed based on the x and y derivatives of the image.*

m) *'p': convert the image to grayscale and plot the gradient vectors of the image every N pixel and let the plotted gradient vectors have a length of K. Use a track bar to control N. Plot the vectors as short line segments of length K.*

n) *'r': convert the image to grayscale and rotate it using an angle of theta degrees. Use a track bar to control the rotation angle.*

o) *'h': display a short description of the program, its command line arguments, and the keys it supports.*

2. PROPOSED SOLUTION

2.1. THE USER IS ASKED TO WRITE A FILENAME TO LOAD AN IMAGE TO PERFORM OPERATIONS ON IT OR PRESS ENTER KEY TO CAPTURE AN IMAGE FROM THE CAMERA.

2.2. IF THE IMAGE IS 1 CHANNEL ONLY (IT IS GRAY) IT IS CONVERTED TO 3 CHANNEL IMAGES.

2.3. IF THE IMAGE IS TOO BIG TO FIT THE SCREEN IT IS REDUCED UNTIL IT FITS IT.

2.4. SPECIAL KEYS FUNCTIONS:

a) To reload the image “reloadimage(image)” function is called. It reloads the original image if it was from a file or takes another picture from the camera if the path name it is not specified.
b) The image is saved to “output.jpg” using openCV function “imwrite()”.
c) The image is converted to grayscale using openCV function “cvtColor()”
d) The image is converted to grayscale converting each pixel to gray from the bgr values using the formula: g = 0.299*b + 0.587*g + 0.114*r
e) To cycle through the 3 channels bgr each time the key is pressed, two of the channels are seted to zero.
f) The image is converted to gray using openCV function “cvtColor()”. Then a track bar that controls a smoothing filter is created in the window to control the amount of smoothing.
g) I tried to convolve a filter with the image creating a function that performed convolution “conv” but I was not getting it right completely. Some errors appeared when trying to multiply and adding the image with the kernel (problems of sizing). Finally, I fixed with the help of a function from the internet “convolution” which was very similar to what I tried to implement.
h) To downsample the image without smoothing its dimensions are divided by 2 with “resize()” function.
i) To downsample the image with smoothing the openCV function “pyrDown()” is used.
j) The image is converted to grayscale as in c). Then a Sobel filter of size 5 is applied to image in x direction. Finally, it is normalized.
k) The image is converted to grayscale as in c). Then a Sobel filter of size 5 is applied to image in y direction. Finally, it is normalized.
l) Both Sobel filters in x and y directions are applied. Then the magnitude is calculated as the square root of the sum of them to the power of 2. Finally, the magnitude is normalized afterwards.
m) Image is converted to grayscale. Then a Sobel filter is applied in both directions. Continuously, for every N pixel the angle of its gradient is calculated as the inverse of the tangent of “sobely” divided by “sobelx”. Finally, knowing the angle, both x and y components can be calculated and the arrows drawn.
n) Image is converted to grayscale. Then it rotate through the track bar using openCV functions “getRotationMatrix2D()” and “warpAffine()”.
o) Information about keys functionally is print on terminal.

3. IMPLEMENTATION DETAILS

The implementation of the program has been done in a Mac OS operated system. The installation of OpenCV was pretty straight forward as the system already had python 3.

During the implementation / programming stage I had several problems:

First, with global and local variables declaration. I got some errors of variables that were not declared. However, debugging with the terminal I was able to fix them.

Second, about OpenCV functions. They were very straight forward to use.

Implementing the own function that converts the image to grayscale I had some problems because I was getting a white image when it was used. The mistake was in the type of array that for images needs to be type uint8 and it was type float (because of the dot product function of numpy). Once it was converted to uint8 the problem was solved.

Also, I had some issues as well with the track bar because I was not reloading the image correctly or passing the parameters in a wrong way. However, once I discover the problem it was the same solution for all the functions that used the track bar.

When I tried to implement the convolution function of g) I had many problems. I was not getting the proper sizes when doing operations with the matrixes after making the padding. At the end I found an implemented function on the internet “convolve” which was very similar to what I did “conv” (almost the same). First, I use this function from the internet which I had to modify a little. First, it did not accept kernels with even dimensions, so the N passed has to be odd. And second the N cannot be too large because of the time processing (it takes much longer than the implementation with OpenCV). I changed so the greatest N is equal to 20 (actually 21 because I am passing only odd numbers). Finally once I figurate this problems: odd kernel dimensions, N not too big and the right size of the operation matrixes I fixed and used my implementation “conv”.

For both cases, file loaded or the camera photo, I supposed that for each key pressed the image was reloaded again. So, for the case of camera photo the image will variate each time one of the multiple option keys are pressed as it reloads/takes a new photo from the camera.

Finally, I tried to make the track bar disappear for those methods that did not use it if they were called after calling one that uses it. However, I could not do it. A possible solution could be to create two windows, one without the track bar and another one with it (if the track bar was used). However, since the track bar does not bother much for the plotting images, this solution has not been implemented.

4. RESULTS AND DISCUSSION

Below I am going to show the output of all the functions:

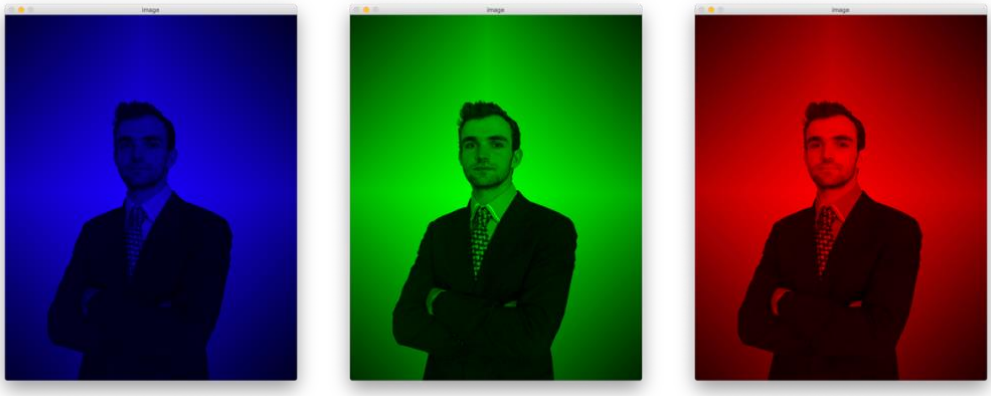
a) This is the original image and it is the one that is reloaded when 'i' key is pressed.

b) All images used in this document have been exported from the program using the key 'w' that is the one that saves the image to a file name "outpu.jpg".

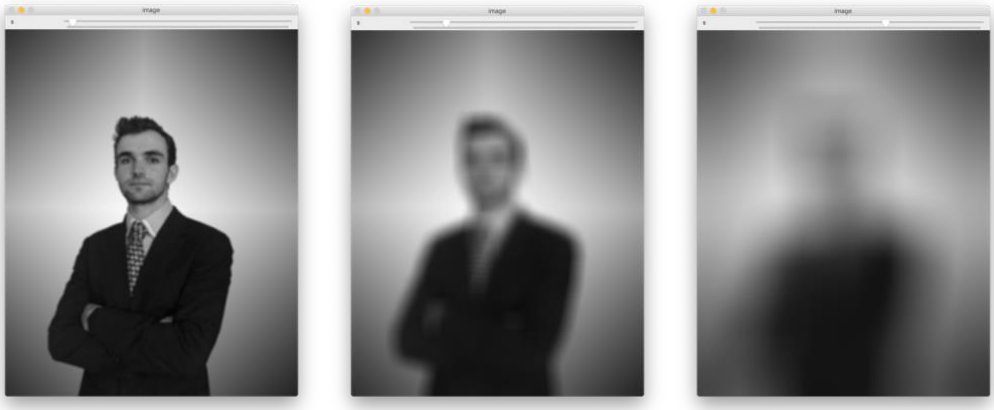
c) When 'g' key is pressed the image is converted to gray using the openCV function.

d) When 'G' is pressed the same conversion to gray is done but using my own code instead of openCV function. The result is indistinguishable from the previous one and takes around the same time for execution.

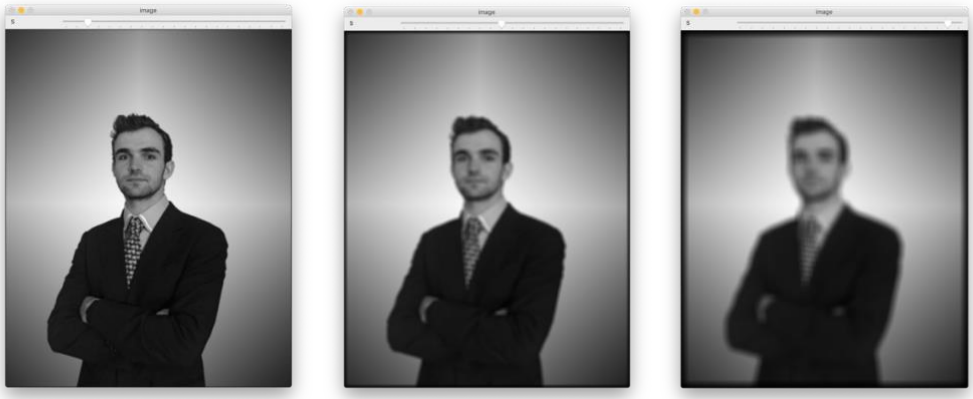
e) When ‘c’ is pressed several times, the image displays the 3 channel colors bgr.



f) When ‘s’ is pressed image is converted to gray and smooth is controlled with the track bar.



g) When ‘S’ is pressed image is converted to gray and smooth is controlled with the track bar using my own method.



h) When ‘d’ is pressed the image is downsampled by 2 without smoothing. Since I am reducing the size of the image I cannot show that it is downsample by 2.

i) When ‘D’ is pressed the image is downsampled by 2 with smoothing. Again, I cannot show the downsample but the smoothing can be seen in the image compared to the previous one.

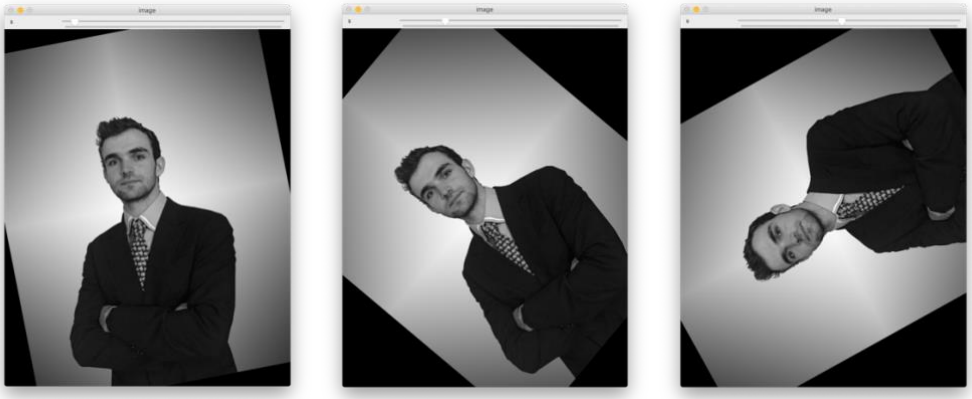
j) When ‘x’ is pressed the image is converted to gray and convolve with an x derivative filter (Sobel).

k) When ‘y’ is pressed the image is converted to gray and convolve with an x derivative filter (Sobel).

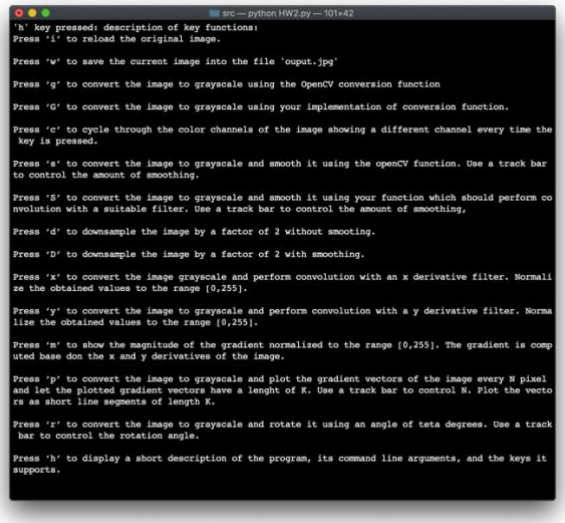
l) ‘m’: show the magnitude of the gradient normalized to the range [0,255]. The gradient is computed based on the x and y derivatives of the image.

m) ‘p’: convert the image to grayscale and plot the gradient vectors of the image every N pixel and let the plotted gradient vectors have a lenght of K. Use a track bar to control N. Plot the vectors as short line segments of length K.

n) ‘r’: convert the image to grayscale and rotate it using an angle of theta degrees. Use a track bar to control the rotation angle.



o) Print help options



5. REFERENCES

https://stackoverflow.com/questions/3614163/convert-single-channle-image-to-3-channel-image-c-opencv

https://stackoverflow.com/questions/43373521/how-to-do-convolution-matrix-operation-in-numpy

https://stackoverflow.com/questions/30079740/image-gradient-vector-field-in-python

http://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/sobel_derivatives/sobel_derivatives.html

https://stackoverflow.com/questions/13003949/faster-way-to-loop-through-every-pixel-of-an-image-in-python

http://docs.opencv.org/3.3.0/da/d6e/tutorial_py_geometric_transformations.html

http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_core/py_basic_ops/py_basic_ops.html

https://cython.readthedocs.io/en/latest/src/tutorial/numpy.html