cs512 Assignment 2 : Sample Report

Karan Bhatiya

Department of Computer Science

Illinois Institute of Technology

1. Problem Statement

The program should be designed to perform simple image manipulation using openCV. First it will load the image by reading it or directly capturing the image from camera. Then we have to apply following special keys on the keyboard to image manipulation:

a. 'i' - reload the image
b. 'w' - save the current image to out.jpg
c. 'g' - convert the current image to grayscale using openCV conversion function
d. 'G' - convert the current image to grayscale using implementation of conversion function
e. 'c' - cycle through the different channel of the image showing different channel image every time we pressed the key
f. 's' - convert the image to grayscale and smooth the image using opencv function and trackbar
g. 'S' - convert the image to grayscale and smooth the image using custom function and trackbar
h. 'd' - downsample the image by factor of 2 without smoothing
i. 'D' - downsample the image by factor of 2 with smoothing
j. 'x' - perform convolution with X derivative filter
k. 'y' - perform convolution with Y derivative filter
l. 'm' - compute gradient based on X and Y derivative of the image
m. 'p' - convert the image to grayscale and plot the gradient vector
n. 'r' - rotate the image by angle theta by converting image into grayscale

2. Proposed Solution

The program should be designed to perform various manipulation on image. To perform the various different manipulation user have to press the special keys of the keyboard as mentioned in the problem statement. The algorithm comprises of loop and a custom function other than inbuilt openCV function.

3. Implementation Details

a) In this we cancelled all the previous operations and reload the original image using **'imshow'** function of openCV.

b) In this we store the current image into out.jpg image for that we used **'imwrite'** function of openCV.

c) To convert the color image to grayscale using inbuilt function of openCV i.e. **'cv2.cvtColor (im, cv2.COLOR_BGR2GRAY)'**. Here **'im'** denotes the original color image and **cv2.COLOR_BGR2GRAY** represents conversion of color to grayscale.

d) To convert the color image to grayscale I defined one function **'converttograyscale'**.
In this function first, I multiply the blue channel of the image with value 0.21, green channel of the image with value 0.72 and red channel of the image with value 0.07. Now to get normalized grayscale image I apply **'grayValue.astype(np.uint8)'**.

e) Firstly, I apply **'split'** function to separate out color channel of the image into r,g,b. To get different color channel of the image I used if else condition.
if k==1
cv2.imshow ('Red',r)
if k==2
cv2.imshow ('Green',g)
if k==3
cv2.imshow ('Blue',b)

f) Step 1: Firstly, I applied **'cv2.cvtColor (im, cv2.COLOR_BGR2GRAY)'** to get the grayscale image of the original image.
Step 2: Then I created trackbar using **'cv2.createTrackbar('S', 'Smoothing Image',3,5,onChange)'**
Step 3: Where S: 'the variable name', 'Smoothing Image': Window Name, '3': starting point of trackbar, '5': end point of trackbar and 'onChange': callback function.
Step 4: Then we get trackbar position of the image using 'cv2.getTrackbarPos ('S', 'Smoothing Image')'.
Step 5: Before applying convolution filter on the image, I checked if the trackbarposition is odd or not using 's%2!=0'
Step 6: If the value of s is odd, I apply the **'cv2.GaussianBlur (grayscale,(s,s),0)'**

g) Apply first 5 step of 'f' as it is.
Step 6: create filter matrix of **s*s** where each value is 1.
Step 7: Apply **'cv2.filter2D (grayscale,-1,kernel)'** to get smoothing image where **'kernel'** is filter matrix.

h) Step1: create blurr image using **'cv2.blur(img,(3,3))' -** Aplied smoothing
Step 2: Apply **'cv.PyrDown (im, 2)'** inbuilt openCV function to get downsampling image using smoothing.

i) Apply **'cv.PyrDown (im, 2)'** inbuilt openCV function to get downsampling image using smoothing.

j) Step1: Apply the sobel convolution filter on X derivative using **'cv2.Sobel (grayscale, cv2.CV_64F, 1, 0, ksize=3)'**
Step2: Normalize the sobel convolution filter image using **'np.uint8(x)'**

k) Step1: Apply the sobel convolution filter on Y derivative using **'cv2.Sobel (grayscale, cv2.CV_64F, 0, 1, ksize=3)'**
Step2: Normalize the sobel convolution filter image using **'np.uint8(y)'**

l) Step 1: Firstly, I applied **'cv2.cvtColor (im, cv2.COLOR_BGR2GRAY)'** to get the grayscale image of the original image.
Step 2: Apply the laplacian convolution filter along X and Y derivative using **'cv2.Laplacian (grayscale, cv2.CV_64F, ksize=3)'**

m) Step 1: Firstly, I applied **'cv2.cvtColor (im, cv2.COLOR_BGR2GRAY)'** to get the grayscale image of the original image.
Step 2: Apply the laplacian convolution filter along X and Y derivative using **'cv2.Laplacian (grayscale, cv2.CV_64F, ksize=5)'**
Step 3: Apply the sobel convolution filter on X derivative using **'cv2.Sobel (grayscale, cv2.CV_64F, 1, 0, ksize=5)'**
Step 4: Apply the sobel convolution filter on Y derivative using **'cv2.Sobel (grayscale, cv2.CV_64F, 0, 1, ksize=5)'**
Step 5: Plot the result in 4 windows using **matplotlib** opencv library.

n) Step 1: Firstly, I applied **'cv2.cvtColor (im, cv2.COLOR_BGR2GRAY)'** to get the grayscale image of the original image.
Step 2: Then I created trackbar using **'cv2.createTrackbar('S', 'Smoothing Image',0,180,onRotate)'**
Where S: 'the variable name', 'Smoothing Image': Window Name, '0': starting point of trackbar, '180': end point of trackbar and 'onRotate: callback function.
Step 3: Then we get trackbar position of the image using **'cv2.getTrackbarPos ('S', Rotational Image')'**.
Step 4: Apply **'cv2.getRotationMatrix2D ((col/2, row/2),r,1)'** inbuilt openCV function
Step 5: Apply **'cv2.warpAffine (grayscale,rotate,(col,row))'** inbuilt openCV function

Problems Faced:

1. As we know, we can't use even matrix because it's not symmetric. So, it throwing error when trackbar is on even number. To overcome this problem I had implemented logic of n%2!= 0. So it only works if it's odd.
2. One more problem I'm facing is that waitKey() not accepting capital alphabets. So please check those function by using different press key of alphabets.

4. Results and Discussion

a. ‘i’ - reload the image

Original Image after cancelling all the operation.

b. ‘w’ - save the current image to out.jpg

Output Image after writing the original image into out.jpg

c. ‘g’ - convert the current image to grayscale using openCV conversion function

Grayscale image of the original Image USING inbuilt openCV function.

d. ‘G’ - convert the current image to grayscale using implementation of conversion function

Grayscale image of the original Image USING inbuilt custom function.

e. ‘c’ - cycle through the different channel of the image showing different channel image every time we pressed the key

Red Channel Image

Green Channel Image

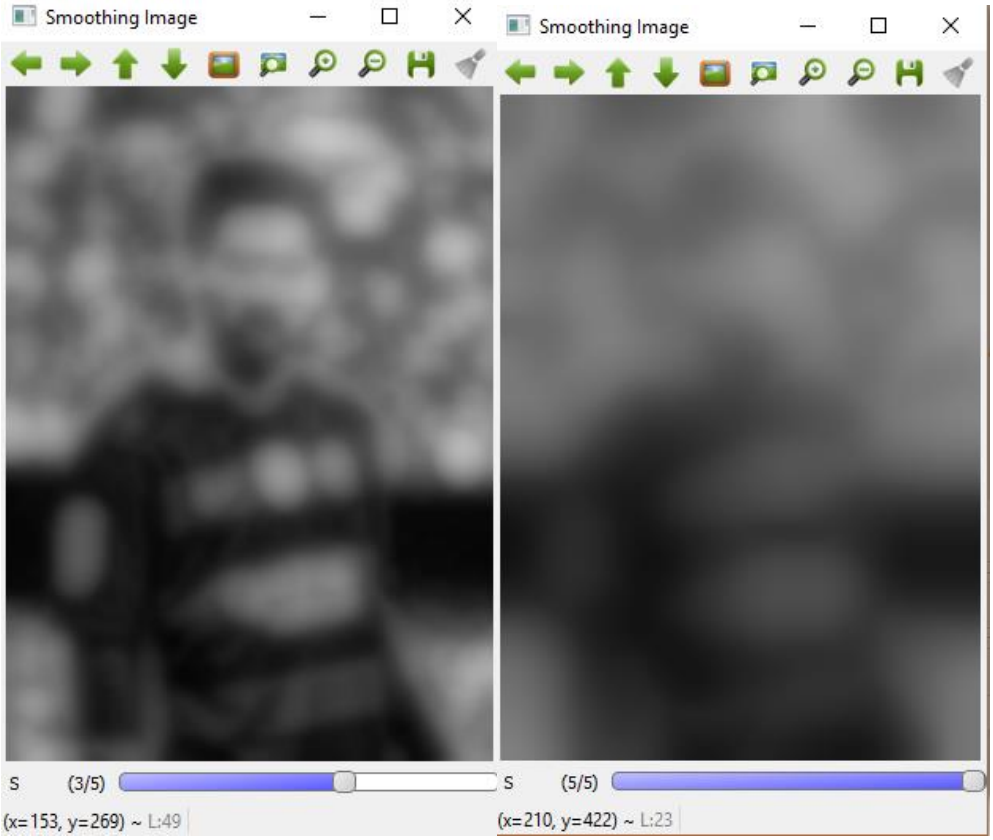
Blue Channel Image

f. ‘s’ - convert the image to grayscale and smooth the image using opencv function and trackbar

Smoothing Image after 3*3 gaussian filter

Smoothing Image after 5*5 gaussian filter

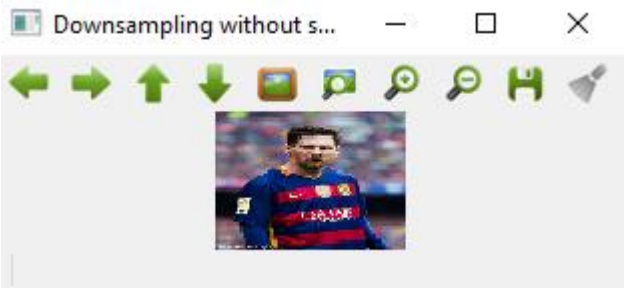
g. ‘S’ - convert the image to grayscale and smooth the image using custom function and trackbar



Smoothing of the image using custom filtration

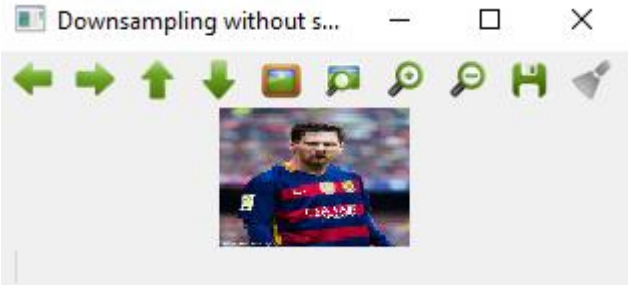
3*3 matrix 5*5 matrix

h. ‘d’ - downsample the image by factor of 2 without smoothing



Downsampling of the image by factor 2 without smoothing

i. ‘D’ - downsample the image by factor of 2 with smoothing



Downsampling of the image by factor 2 with smoothing

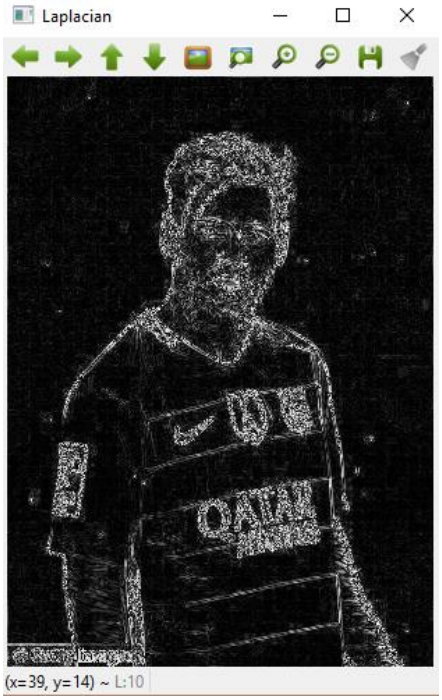
j. ‘x’ - perform convolution with X derivative filter

Result of the image after applying convolution with X derivative filter

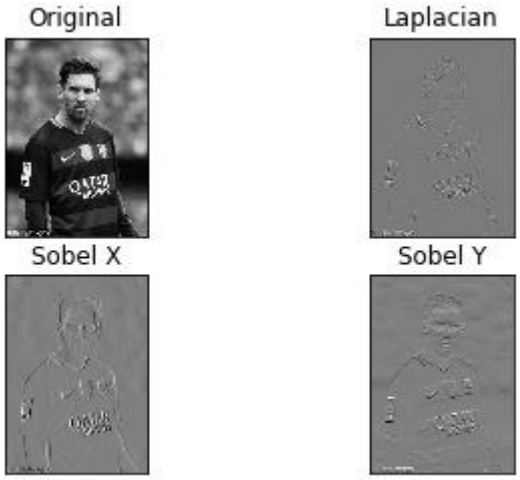
k. ‘y’ - perform convolution with Y derivative filter

Result of the image after applying convolution with Y derivative filter

l. ‘m’ - compute gradient based on X and Y derivative of the image



m. ‘p’ - convert the image to grayscale and plot the gradient vector



n. ‘r’ - rotate the image by angle theta by converting image into grayscale

Result of the image after rotating it by 180.

Rotation of the image after rotating it by 90.

5. References

- https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_image_display/py_image_display.html
- https://extr3metech.wordpress.com/2012/09/23/convert-photo-to-grayscale-with-python-opencv/
- https://stackoverflow.com/questions/51285593/converting-an-image-to-grayscale-using-numpy
- https://stackoverflow.com/questions/10431919/downsampling-without-smoothing
- https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_gui/py_trackbar/py_trackbar.html
- https://docs.opencv.org/2.4/modules/imgproc/doc/filtering.html?highlight=pyrdown#pyrdown
- https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_gradients/py_gradients.html
- https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_geometric_transformations/py_geometric_transformations.html
- https://media.readthedocs.org/pdf/opencv-python-tutroals/latest/opencv-python-tutroals.pdf