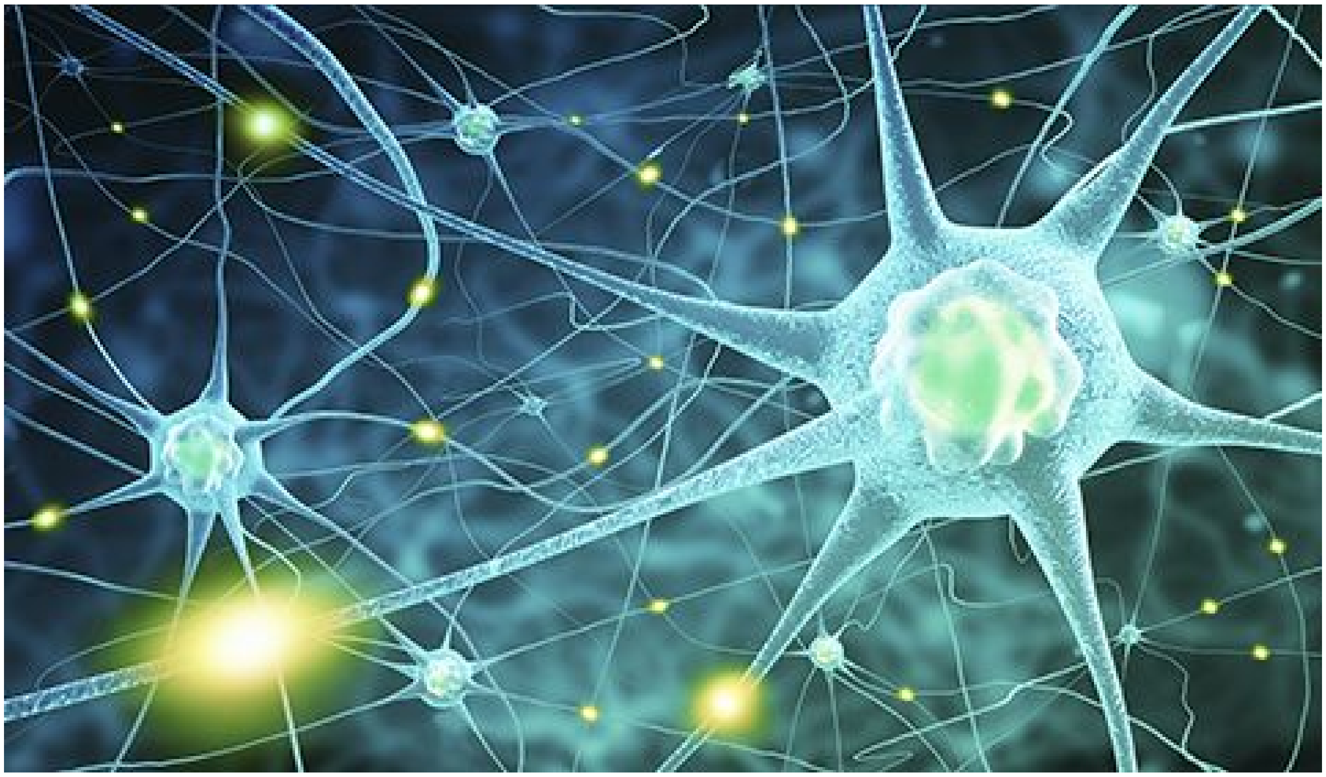
Assignment 1

Computer Vision - CS512



Akshay R

A20442409

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 'out.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 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. There should be inverse mapping.

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

Proposed Solution:

The main method calls various functions to perform different activities when a keystroke is registered. We first input the image to be processed either by a file path or by using the camera. A preliminary check of the size of the image and converting the image to RGB channels will ensure the image can be processed to perform various actions.

To convert the image into a grayscale image we apply the following formula.

Luminosity = 0.299*B + 0.587*G + 0.114*R

Every special key is mapped to individual functions that in turn perform the required job functions. This mapping is through an infinitely running if else statement that breaks only when the escape key is pressed.

Implementation and methodology:

a) ‘i’: reload the original image. - This is implemented through a function called **reload_image**. Each time I reload the image the input is either taken from the path entered or a new image is taken from the camera.

b) ‘w’: save the current image into the file ‘out.jpg.’ - This is implemented through a function called **savefile**. The output is saved as out.jpg

c) ‘g’: convert the image to grayscale using the OpenCV conversion function. - This is implemented through a function called **togrey**. We use the function, cv2.cvtColor.

d) ‘G’: convert the image to grayscale using your implementation of conversion function.- This is implemented through a function called **togreyown**. The formula to convert the color to grayscale is as discussed in the proposed solution.

e) ‘c’: cycle through the color channels of the image showing a different channel every time the key is pressed. - This is implemented through a function called **changecolor**. Count decides what color channel must be displayed next.

f) ‘s’: convert the image to grayscale and smooth it using the openCV function. Use a track bar to control the amount of smoothing. - This is implemented through a function called **smooth**. cv2.filter2D is used to smoothen the image.

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. - This is implemented through a function called **smoothown**. This method calls the conv function which performs convolution.

h) ‘d’: downsample the image by a factor of 2 without smoothing. - This is implemented through a function called downsample. I have implemented this by using the cv2.resize function.

i) ‘D’: downsample the image by a factor of 2 with smoothing. - This is implemented through a function called **downsample_smooth**. I have implemented this by using the cv2.pyrDown function.

j) ‘x’: convert the image grayscale and perform convolution with an x derivative filter. Normalize the obtained values to the range [0,255]. - This is implemented through a function called **convdevnorm**. This function performs convolution with a x derivative filter when the variable xory is set to 1.

k) ‘y’: convert the image to grayscale and perform convolution with y derivative filter. Normalize the obtained values to the range [0,255]. - This is implemented through a function called **convdevnorm**. This function performs convolution with a y derivative filter when the variable xory is set to 0.

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.- This is implemented through a function called **gradient_magnitude**. It uses sobel filters to compute x and y derivatives and then computing the magnitude of the gradient.

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. - This is implemented through a function called **gradient_vectors**. First we compute the sobel x and y filters. We then compute the gradient x and y projections using x + n cos(theta) and y + n sin(theta), n being the number of gradients that is required which is set using a sliding bar. Every gradient is displayed using cv2.arrowedLine() function.

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. There should be inverse mapping. -This is implemented through a function called **rotate**. We use cv2.getRotationMatrix2D to obtain the rotation matrix and then cv2.warpAffine which uses a 2x3 transformation matrix as its input and multiplies it to image matrix to obtain a rotated image.

o) ‘h’: display a short description of the program, its command line arguments, and the keys it supports. - This is implemented through a function called **help.** Contains multiple print statements to show key bindings.

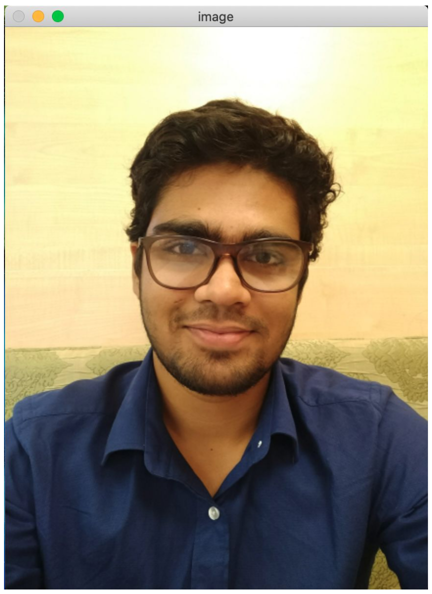
User keystrokes are monitored using an infinitely running while loop. The loop waits for the key press using the function cv2.waitKey() function. An “Escape” key results in closing the application and is implemented using cv2.destroyAllWindows(). The main method is called to execute the program.

Difficulties faced: 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.

The track bar used stacks over each other once called. This was due to calling the trackbar in every function that used it. This problem was not solved and results in multiple track bars when respective functions are called.

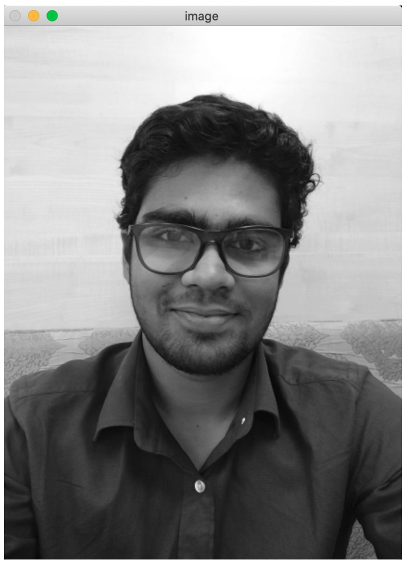
Results and discussion:

a) Every time i is pressed this image is reloaded. This is an image that is extracted through a path where the image can be found.

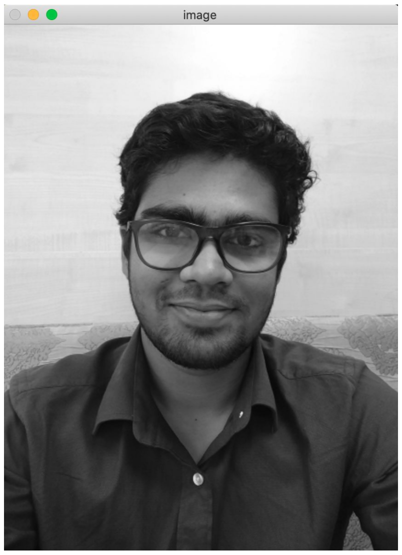


b) ‘w’: save the current image into the file ‘out.jpg.’ - When the w key is pressed the current image is saved as out.jpg to the path where the notebook is saved.

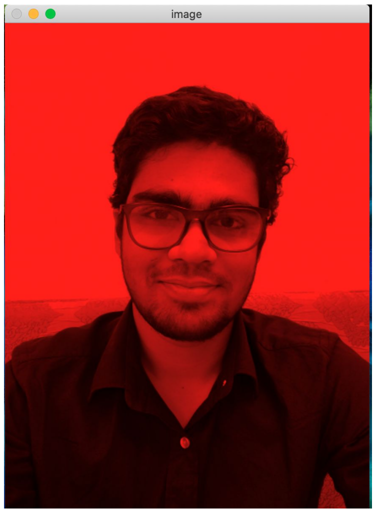
c) ‘g’: convert the image to grayscale using the OpenCV conversion function. Result:

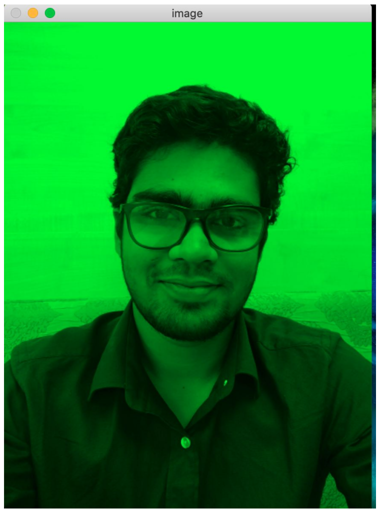


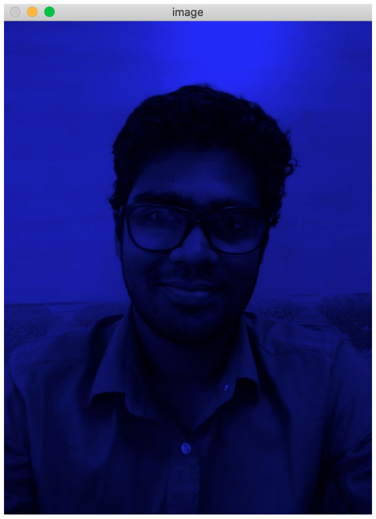
d) ‘G’: convert the image to grayscale using your implementation of conversion function.



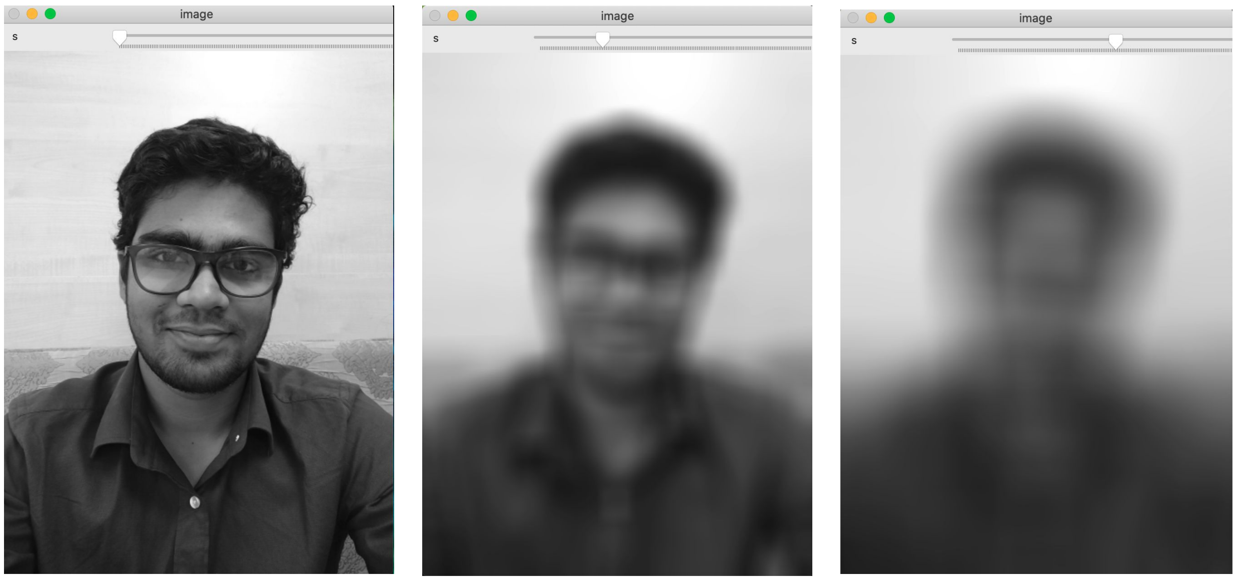
e) ‘c’: cycle through the color channels of the image showing a different channel every time the key is pressed.



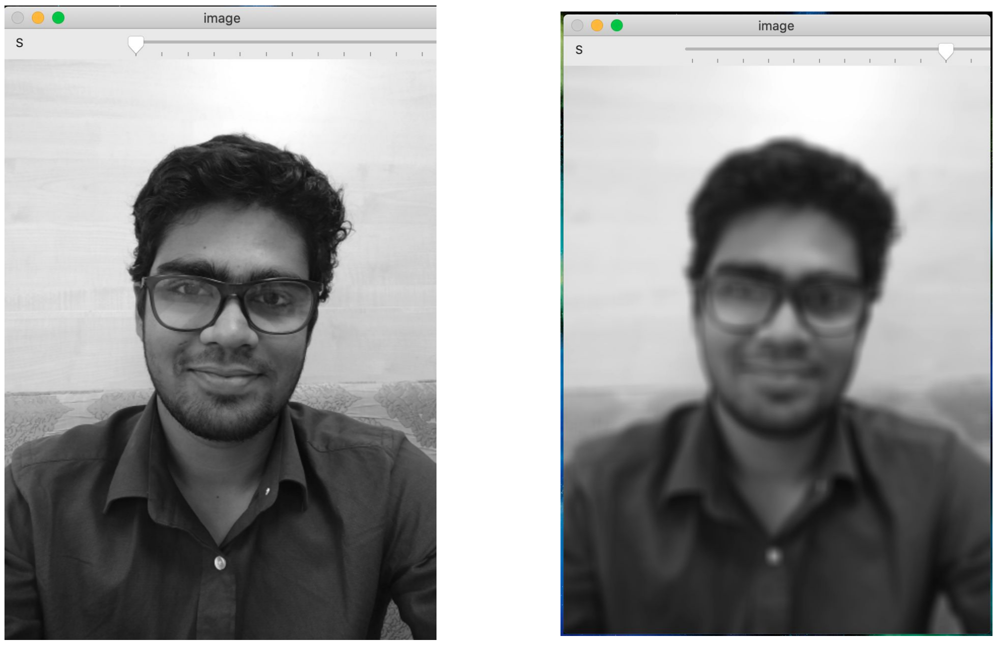




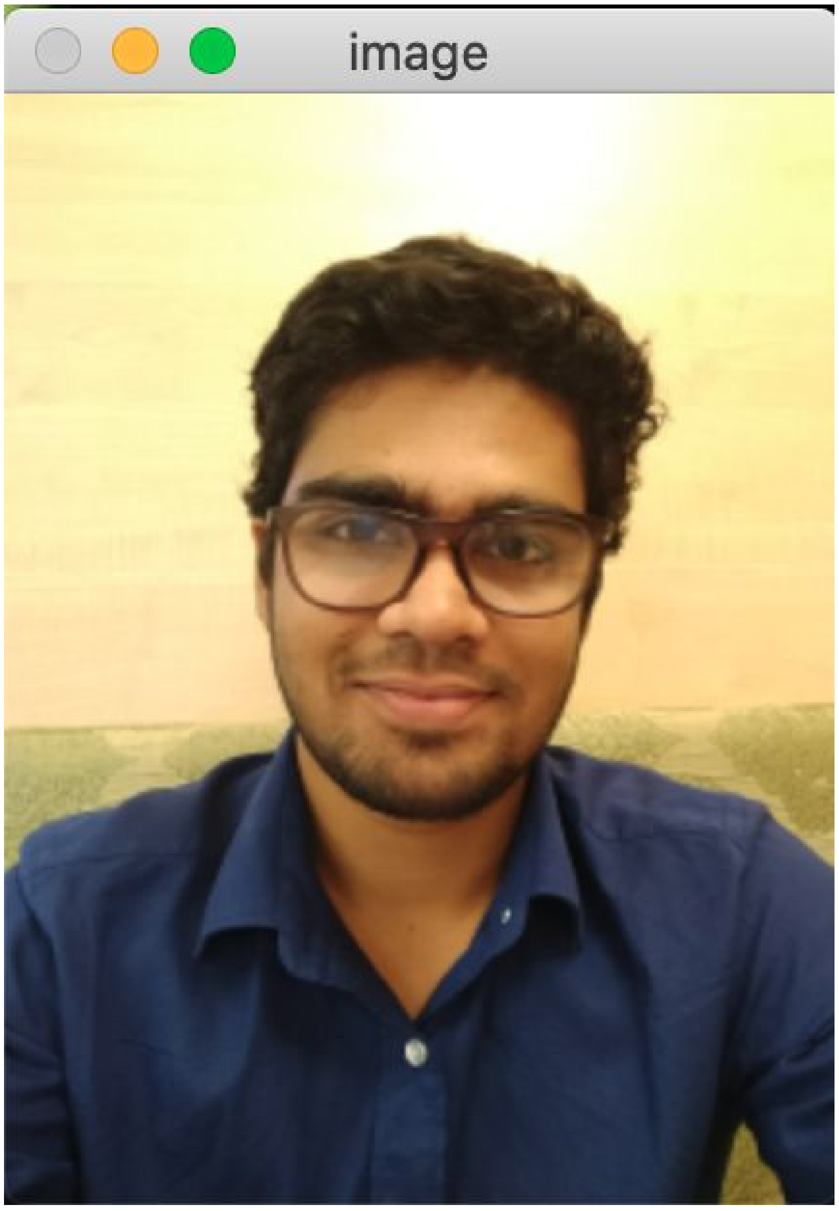
f) ‘s’: convert the image to grayscale and smooth it using the openCV function. Use a track bar to control the amount of smoothing.



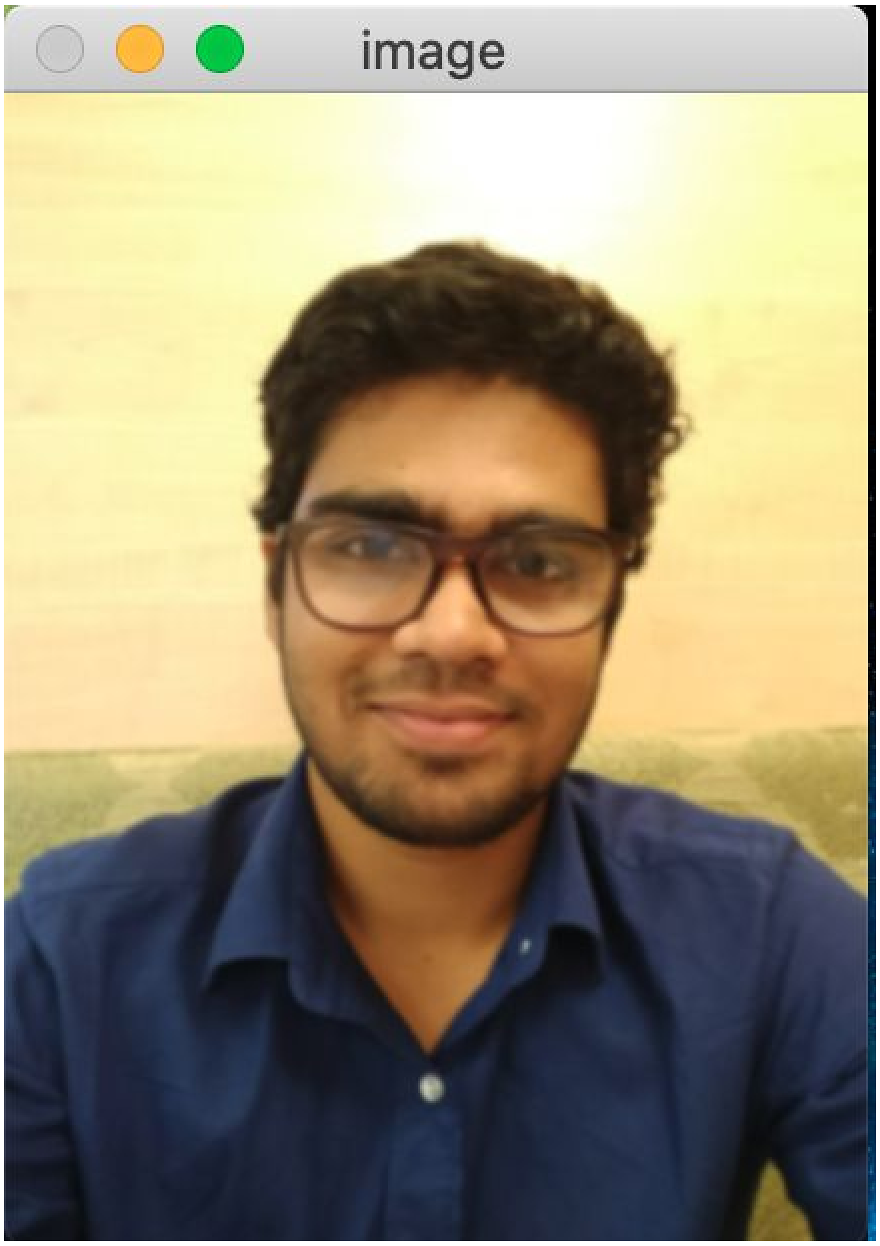
g) ‘S’: convert the image to grayscale and smooth it using your function which should perform convolution with a suitable filter. Use a track bar to control the amount of smoothing.



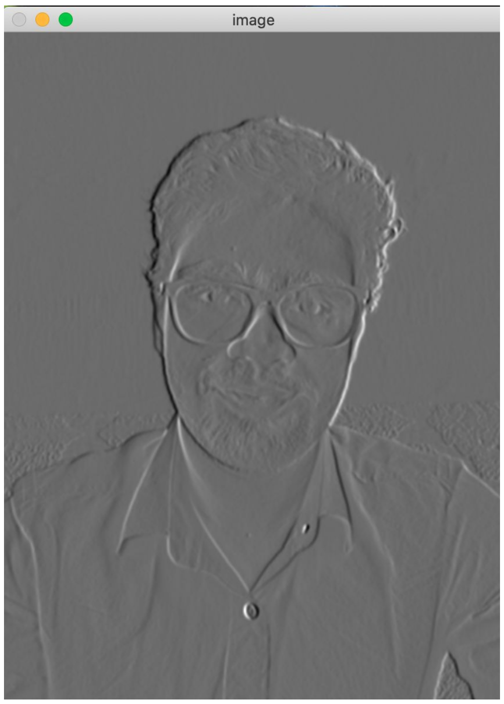
h) ‘d’: downsample the image by a factor of 2 without smoothing. The image is downsampled. This can be observed here as the title bar is appearing bigger than the previous images.



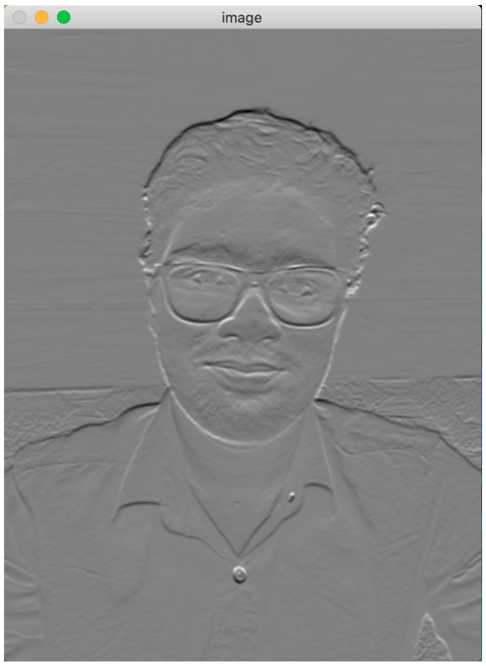
i) ‘D’: downsample the image by a factor of 2 with smoothing.



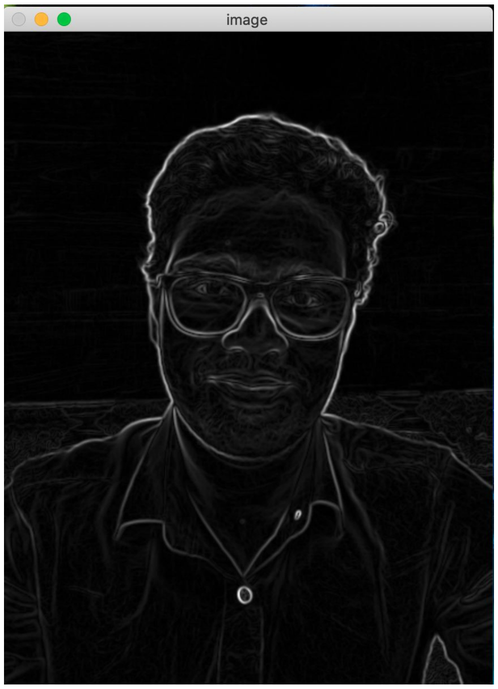
j) ‘x’: convert the image grayscale and perform convolution with an x derivative filter. Normalize the obtained values to the range [0,255]. - **Vertical edges**



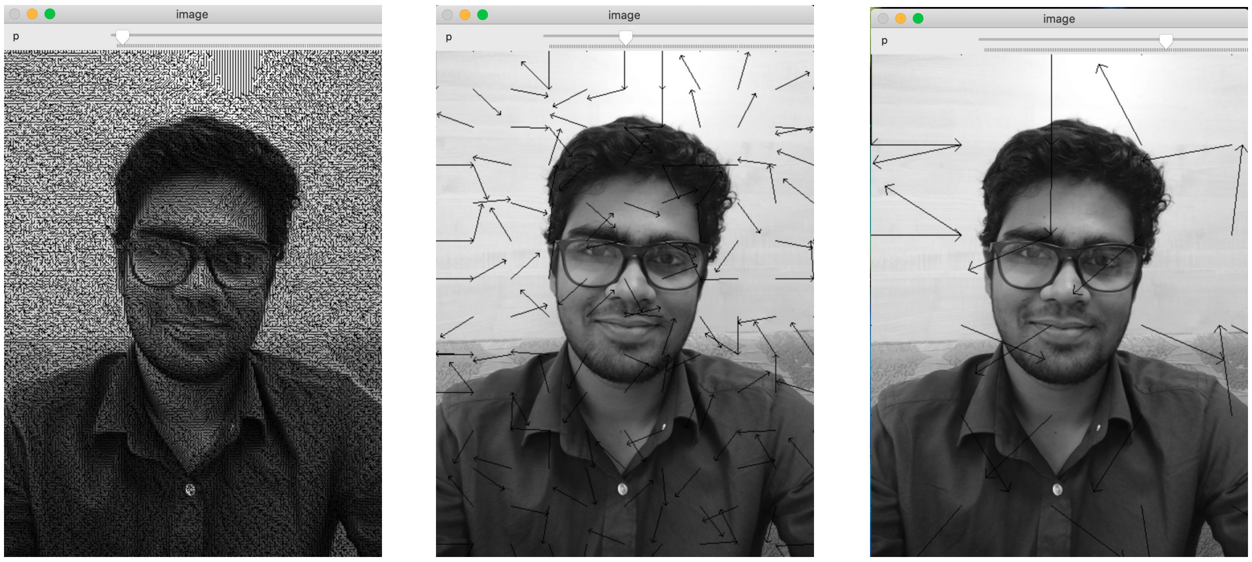
k) ‘y’: convert the image to grayscale and perform convolution with y derivative filter. Normalize the obtained values to the range [0,255]. - **Horizontal edges**



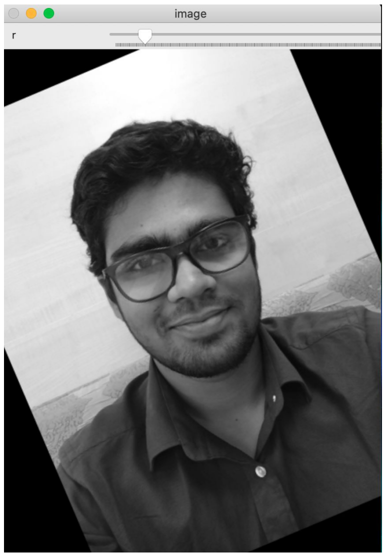
l) ‘m’: show the magnitude of the gradient normalized to the range [0,255]. The gradient is computed based on the x and y derivatives of the image.

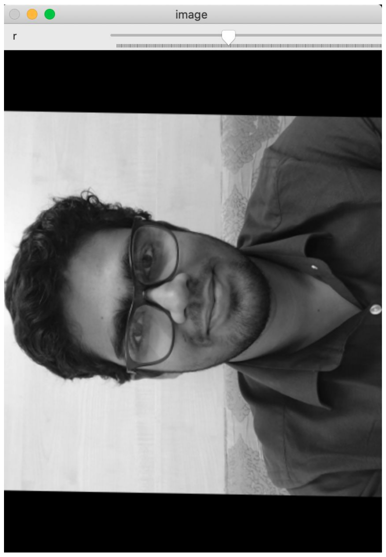


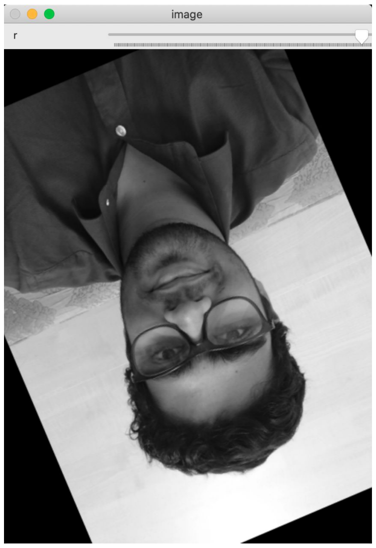
m) ‘p’: convert the image to grayscale and plot the gradient vectors of the image every N pixel and let the plotted gradient vectors have a length of K. Use a track bar to control N. Plot the vectors as short line segments of length K.



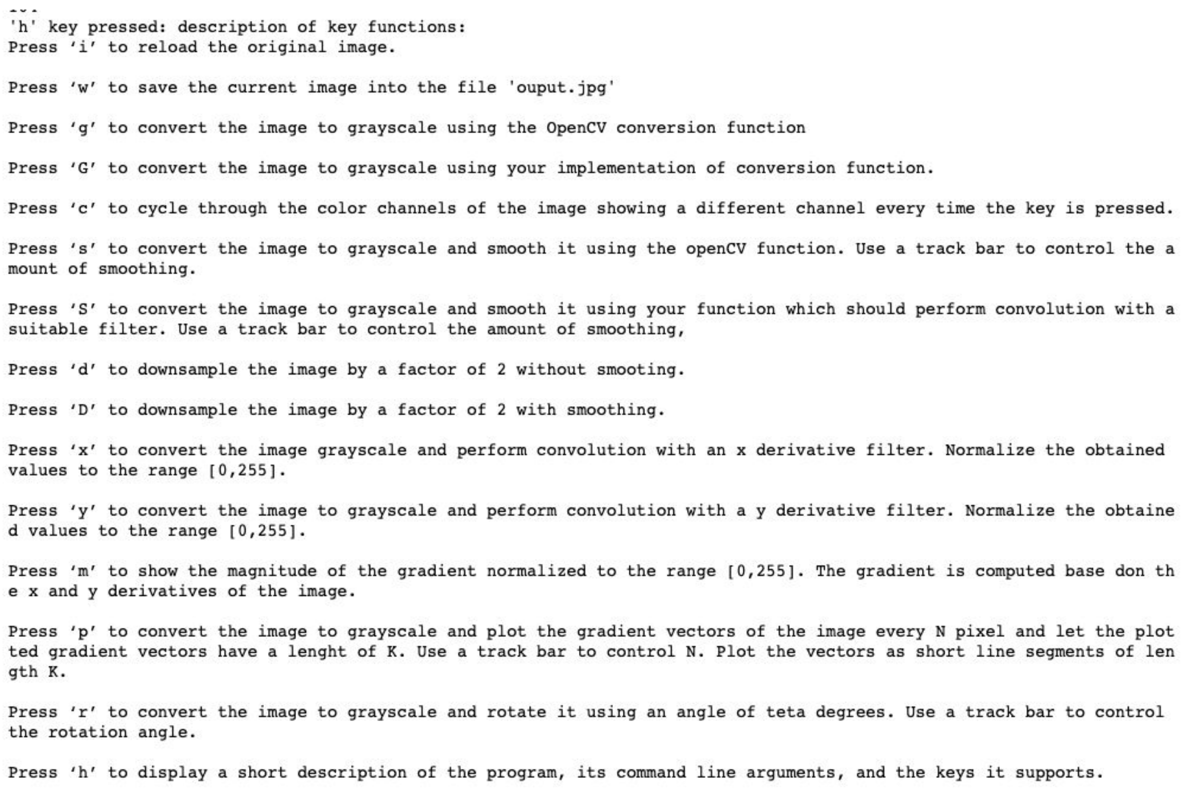
n) ‘r’: convert the image to grayscale and rotate it using an angle of theta degrees. Use a track bar to control the rotation angle. There should be inverse mapping.







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



```
'h' key pressed: description of key functions:
Press ‘i’ to reload the original image.

Press ‘w’ to save the current image into the file 'ouput.jpg'

Press ‘g’ to convert the image to grayscale using the OpenCV conversion function

Press ‘G’ to convert the image to grayscale using your implementation of conversion function.

Press ‘c’ to cycle through the color channels of the image showing a different channel every time the key is pressed.

Press ‘s’ to convert the image to grayscale and smooth it using the openCV function. Use a track bar to control the a
mount of smoothing.

Press ‘S’ to 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,

Press ‘d’ to downsample the image by a factor of 2 without smooting.

Press ‘D’ to downsample the image by a factor of 2 with smoothing.

Press ‘x’ to convert the image grayscale and perform convolution with an x derivative filter. Normalize the obtained
values to the range [0,255].

Press ‘y’ to convert the image to grayscale and perform convolution with a y derivative filter. Normalize the obtaine
d values to the range [0,255].

Press ‘m’ to show the magnitude of the gradient normalized to the range [0,255]. The gradient is computed base don th
e x and y derivatives of the image.

Press ‘p’ to convert the image to grayscale and plot the gradient vectors of the image every N pixel and let the plot
ted gradient vectors have a lenght of K. Use a track bar to control N. Plot the vectors as short line segments of len
gth K.

Press ‘r’ to convert the image to grayscale and rotate it using an angle of teta degrees. Use a track bar to control
the rotation angle.

Press ‘h’ to display a short description of the program, its command line arguments, and the keys it supports.
```

References:

- https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_core/py_basic_ops/py_basic_ops.html
- https://docs.opencv.org/3.3.0/da/d6e/tutorial_py_geometric_transformations.html
- https://docs.opencv.org/2.4/doc/tutorials/imgproc/imgtrans/sobel_derivatives/sobel_derivatives.html