CS 512 : Assignment: 2 Report

By:Charu Saxena
Department of Computer Science :IIT
9th october,2017

Abstract

In this report we will discuss about various implementation and usage of various opencv functions and python functions using python 3.6. Some of the implementations and usage discussed are basic computer vision and image processing problems including capturing images and live images, modifications on image.

1.Problem Statement

To write a program to input an image from file or by camera directly and should have specifications as stated below:

1. Read image file through command line or through capture from camera which is to be continuously processed.
2. The read image should be read as a 3 channel color image.
3. The program should work with any image size.
4. Special keys to provide different modification on the image:listed in(http://www.cs.iit.edu/~agam/cs512/share/cs512-hw2.pdf)

2.Proposed Solutions

I have used some of the built-in functions for opencv in python for some of the questions whereas, in others I have made functions for certain questions where we were asked to implement the certain openCV functions listed as:

1. Gray Scale function
2. S-convert to grayscale,Smooth it using a filter using convolution and use a track bar to control the smoothing.
3. Various others involved using built-in functions listed :
    a. Reloading by loading the original image using using opencv function cv.imread()
    b. Save it using using opencv function imwrite()
    c. Greyscale is done using opencv function cvtColor(img, cv.COLOR_BGR2GRAY)
    d. Gray scale for own implementation using average of all colors and applying, this is further discussed in implementation.

e. To cycle the color the color channel using flags and using opencv function imshow()

f. Converting to grayscale and smooth it using track bar using opencv function createTrackbar()further given in implementation

g. Converting to grayscale and smooth it using convolution of own function using track bar using opencv function createTrackbar()further given in implementation

h. Downsample by 2 using opencv function pyrDown

i. Upsample by 2 using opencv function pyrUp

j. Convert the image to grayscale and use user made function for convolution

k. Perform convolution on x-derivative of the image using opencv function Sobel(), and normalise it using normalize().

l. Perform convolution on y-derivative of the image using opencv function Sobel(), and normalise it using normalize().

m. Get gradient vectors on the image using matplotlib functions like plt.quiver and numpy using np.meshgrid to plot the vectors.

n. Rotate the image using trackbar using opencv function createTrackbar() and rotate argument

o. Print description of data and keys used, and on the image a name using opencv function putText().

3.Implementation

In the program, I used python and its packages like opencv,numpy,PIL ,matplotlib.

Grayscale: to grey scale the image

```
def gray():
   imge_copy= img.copy()
   for i in range(img.shape[0]):
      for j in range(img.shape[1]):
         average = int(sum(img[i,j])/3)
         imge_copy[i,j] = (average,average,average)
   cv.imshow('image',img)
```

SlideHandler:to insert a side handler for smoothing

```
def slideHandler(i):
    global img
    n=cv.getTrackbarPos('trackbar','image')
    img = cv.filter2D(imge,-1)
    cv.imshow('image',img)
```

rotateHandler: to insert a side bar which when slides to rotate the image

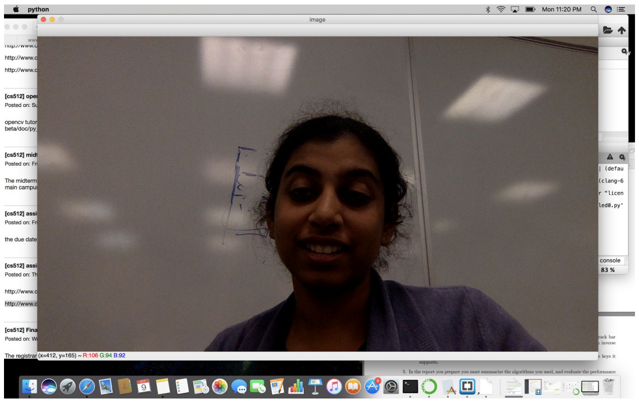
```
def rotateHandler(i):
    global img
    n=cv.getRotationMatrix2D((r/2,c/2),i,1)
    img = cv.warpAffine(imge,n,(c,r))
    cv.imshow('image',img)
```

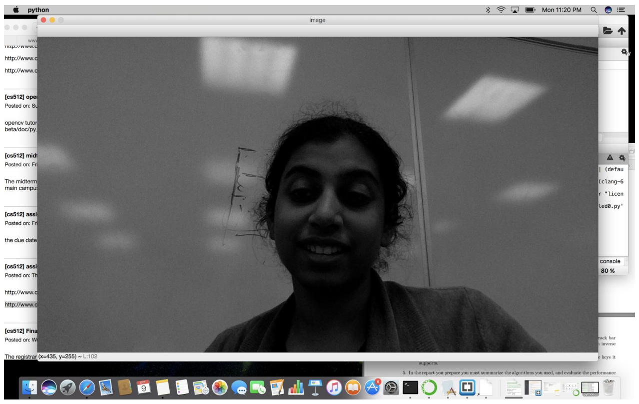
FOR TO IMPLEMENT GRADIENT VECTOR: here first the gradient is found using sobel() And the image size is taken in meshgrid and then the derivatives and the image array values of x and y are taken in quiver abd the plot is show()

```
elif k== ord('p'):
        img = img0
        xval = np.arange(img.shape[0])
        yval = np.arange(img.shape[1])
        x,y = np.meshgrid(xval,yval,indexing='ij')
        hori = cv.Sobel(img,cv.CV_64F,1,0,ksize=5)
        hor=cv.normalize(hori,0,255,cv.NORM_MINMAX)
        veri = cv.Sobel(img,cv.CV_64F,1,0,ksize=5)
        ver=cv.normalize(veri,0,255,cv.NORM_MINMAX)
        u = v = np.zeros((11,11))
        u[0,0]=0.2
        plt.quiver(x,y,u,v,scale=1, units ='width')
       plt.show()
```

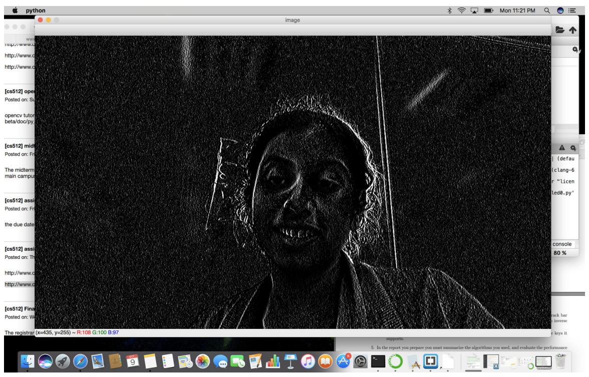
4.Result/Discussion

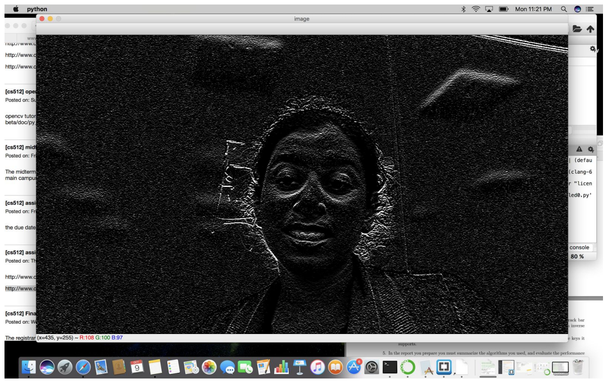
The images as a result of above discussed functions are:



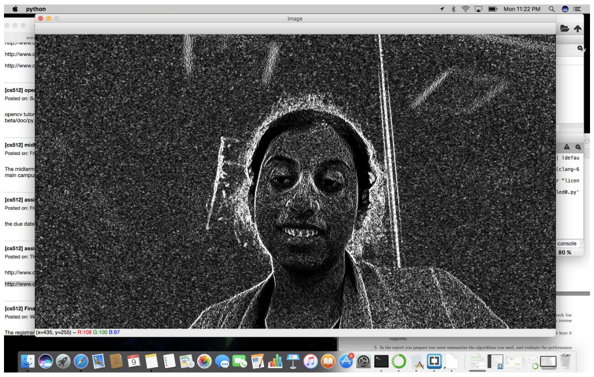


Greyscaled image when key “D” is pressed





For y-derivative **For x-derivative**



For magnitude

As per above results we see that most of the features were implemented successfully.

Here the grey scale the values and parameter used was default values also same applies to all other used.

5.References:

https://docs.scipy.org/doc/numpy-1.13.0/reference/generated/numpy.meshgrid.html

http://docs.opencv.org