cs512 Assignment 2

Megha Tatti
cwid: A20427027
Fall2018

Abstract

We write a program to perform simple image manipulation using OpenCV. The program loads an image by either reading it from a file or capturing it directly from the camera. The input can be an image of a certain generic formats which performs various tasks such as cycling through different color channels, converting it to a grayscale, the original image in different windows with RGB colors. Sampling the images with the implementation smoothing, rotating with respect to an angle at a given axis, translate and scale it up and down, using filters to remove noises in the images and for further enhancing of images.

Problem Statement

The problem I am facing here is in processing the image with a user written program which explicitly takes an image and loads it and then perform various functions such as

- **Load the image**: We load an image from a directory or we have to allow the camera to capture an image and store
- **Saving the file**: Saving of an image to different locations after being read.
- **Gray Scale image**: Converting the original image to different scales - grayscale is among the top functions.
- **Noise filtering** : it's the most crucial part in processing the image, distortion and any amount of grains has to be reduced to optimize our functions
- **Smoothing** – Smoothing is often used to reduce noise within an image.
- **Sampling an image**: Sampling up and down, with and without the smoothing filter.
- **RGB Color gradient and magnitude**: The range of positions of different color pixels in an image is located along with the variation in the intensity of the image.
- **X-derivative:** Retrieves the x-derivatives of the image.
- **Y-derivative:** Retrieves the y-derivates of the image.

Proposed solution:

- Open CV from Anaconda python
- Load the image
- Processing an image after loading and transforming an image to make it best fit for the functions so that it gives an efficient output every time a different size image with different dimensions are given as an input

- Installing different libraries to make the new functions compatible with the existing code such as matplotlib

Algorithm:

The basic algorithm starts with giving an image of certain dimensions and format that is then being altered with respect to different co-ordinate systems.

For a particular image which is either uploaded from the directory or captured live using a webcam, it can be processed using different functionalities as below:

Step 1: start

Step 2: select an option

Step 3: image is uploaded or captured via webcam

Step 4: As per the option selected, the function performing the operation are called

Implementation Details

The implementation for the above proposed problem statement is as follows:
We set the path using the below functions

```
parser=argparse.ArgumentParser()
parser.add_argument("--path",help="path")
args=parser.parse_args()
```

The above functions help us to know where the command passed is saved.

- **Load an image**

  Problem design issues and problems faced:
  Here I was finding it difficult to know how to set the path to where image is be captured and loaded. I didn’t know how the waitKey() function works. Found it difficult to capture the image and display it.
  So, I solved the above problem using the below code implementations and the setting path issue was solved by using the function:

  ```
  parser=argparse.ArgumentParser()
  parser.add_argument("--path",help="path")
  args=parser.parse_args()
  ```

  To use this I imported the argparse() package.

  The implementation for this problem is as follows:

  ```
  if(len(sys.argv)< 2):
  image=cv2.VideoCapture(0)
  check,imge=image.read()
  ```

--These commands helps us capture the image from the camera.

```
cv2.imshow('Capturing',imge)
```

-This command helps us set the frame to display the image captured in the previous step.

```
ker=cv2.waitKey(0)
image.release()
else:
imge=cv2.imread("%s"%(args.path),1)
```

-This command helps us read the image from the path set.

```
cv2.imshow('Image',imge)
```

-This command helps to display the image on the screen

```
ker=cv2.waitKey(0)
```

- **Saving the image to file**

To save the image, the following code is used:

```
if k==ord('w'):
cv2.imwrite("out.jpg",imge)
```

-this imwrite() command helps us save the image captured and save it in the 'out.jpg' file.

- **Gray Scale image**

Problem design issues and problems faced:

The process of Color processing of an image consists of converting the image into grayscale and transforming image, was a little tricky for me as I was not properly aware of how to scale and convert the image from color to grayscale. To overcome that, below code implementations helped me.

```
image_gray=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

-*cvtColor* function is used to change the color of the image and the *COLOR_BGR2GRAY* to change it into gray.

```
cv2.imshow("Image_grayscale",image_gray)
```

Here the imshow() function shows the image into window.

```
g=cv2.waitKey(0)
```

- **RGB Color gradient and magnitude**

Problem design issues and problems faced:

Here I didn't know that the order in cv2 was BGR and not RGB. So, had difficulty in solving that. There were problems setting the other 2 channels to 0.

I solved the above problems faced using the following code implementations:

```
blue_imge=np.copy(imge)
```

-This command helps us load the color “blue” in the image. But this happens only when the other 2 channels are made ‘0’.

```
green_imge=np.copy(imge)
```

- This command helps us load the color “green” in the image. But this happens only when the other 2 channels are made ‘0’.

```
red_imge=np.copy(imge)
```

- This command helps us load the color “red” in the image. But this happens only when the other 2 channels are made ‘0’.

```
blue_imge[:,:,1:]=0
```

-This command helps me set the first and second channels to 0.

```
green_imge[:,:,(0,2)]=0
```

-This command helps me set the first and third channels to 0.

```
#2,3 channels are made 0
red_imge[:,:,:2]=0
```

-This command helps me set the second and third channels to 0.

The following code helps us cycle through the color channels on pressing the key ‘c’.

```
if(ker==ord('c')):
cv2.imshow('Blue Channel',blue_imge)
```

-These commands help cycling to the image in ‘blue’

```
r=cv2.waitKey(0)
```

-This command delays the rendering of images to windows by some ‘n’ milliseconds.

```
if(r==ord('c')):
cv2.imshow('Green Channel',green_imge)
```

-These commands help cycling to the image in ‘green’

```
g=cv2.waitKey(0)
```

-This command delays the rendering of images to windows by some ‘n’ milliseconds.

```
if(g==ord('c')):
cv2.imshow('Red Channel',red_imge)
```

-These commands help cycling to the image in ‘red’

- **Smoothing**

  Problem design issues and problems faced:

  Here I had problem in finding out how to use the scaling factor so as to smooth the image.

It was bit tricky to come out with a solution.
So, I used the sliderHandler() function as below and then implemented the smoothing function.

```
def sliderHandler(n):
global imge
kernel=np.ones((n,n),np.float32)/(n*n)
dist=cv2.filter2D(img,-1,kernel)
cv2.imshow('processed',dist)
```

-These commands show the processed image.

```
if(k==ord('s')):
img=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

-This command converts the image to gray scale.

```
cv2.imshow('blur',imge)
cv2.createTrackbar('s','blur',0,10,sliderHandler)
cv2.waitKey(0)
```

-These commands helps to scale and navigate between blurr to sharp smoothing.

- **Sampling**

By factoring of an image, I am creating an "access" image that is a miniaturized or maximized duplicate of your optical resolution "master" scan, basically a way of compressing the image with minimum loss of pixel data. Here, in the problem, I was asked to downscale an image by a factor of 2 without and with smoothing the pixel, for that, I have used resize function of OpenCV, which was very useful to use but the set output image size

To achieve that, below code is used for **downscale the image by 2 without smoothing**

```
print(img.shape)
```

-printing the original image

```
lower_reso = cv2.resize(img,(int(img.shape[1]/4),int(img.shape[0]/4)))
```

-The resize function is taking the original image and shaping it by dividing the number of rows(img.shape[0])and column(img.shape[1]) by 4 for the factoring the image by 2.

```
print(lower_reso.shape)
```

-Printing the low res image

```
cv2.imshow('Modified_Image',lower_reso)
cv2.waitKey(0)
```

This is code for **downscale the image by 2 with smoothing**

```
print(img.shape)
```

-Printing the original image

```
lower_reso = cv2.pyrDown(img)
```

-Here we are using pyrDown instead of resize Because pyrDown not only downsamples it but also blurs an image which is required for the action of smoothing

```
print(lower_reso.shape)
```

-Printing the low res image

```
factorBy4=cv2.pyrDown(lower_reso)
```

-using pyrDown will scale down the image only by factor 1 so applying it again to downsample the image to factor 2.

```
print(factorBy4.shape)
cv2.imshow('factorBy4',factorBy4)
cv2.imshow('Modified_Image',lower_reso)
```

- Printing and showing both the images.

```
cv2.waitKey(0)
```

- **Filtering**

```
allocate outputPixelValue[image width][image height]
allocate window[window width * window height]
edgex := (window width / 2) rounded down
edgey := (window height / 2) rounded down
for x from edgex to image width - edgex
for y from edgey to image height - edgey
i = 0
for fx from 0 to window width
for fy from 0 to window height
window[i] := inputPixelValue[x + fx - edgex][y + fy - edgey]
 i := i + 1
 sort entries in window[]
  outputPixelValue[x][y] := window[window width * window height / 2]
```

- **X-derivative**

The following code implementations helped me get the x-derivative of the image.

```
if (ker==ord('x')):
image_gray=cv2.cvtColor(imge,cv2.COLOR_BGR2GRAY)
sobelx = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]], dtype = np.float)
sobely = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]], dtype = np.float)
gx = cv2.filter2D(image_gray, -1, sobelx)
```

```
gy = cv2.filter2D(image_gray, -1, sobely)
g = np.sqrt(gx * gx + gy * gy)
g *= 255.0 / np.max(g)
cv2.imshow('x_derivative',gx)
```

-These commands helps us retrieve the x-derivative of the image.

```
g=cv2.waitKey(0)
cv2.destroyAllWindows()
```

- **Y-derivative**

```
if (ker==ord('y')):
image_gray=cv2.cvtColor(imge,cv2.COLOR_BGR2GRAY)
sobelx = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]], dtype = np.float)
sobely = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]], dtype = np.float)
gx = cv2.filter2D(image_gray, -1, sobelx)
gy = cv2.filter2D(image_gray, -1, sobely)
g = np.sqrt(gx * gx + gy * gy)
g *= 255.0 / np.max(g)
cv2.imshow('y_derivative',gy)
```

-This helps to normalize output to fit the range. This command helps to get the x-derivative of the image.

```
g=cv2.waitKey(0)
cv2.destroyAllWindows()
```

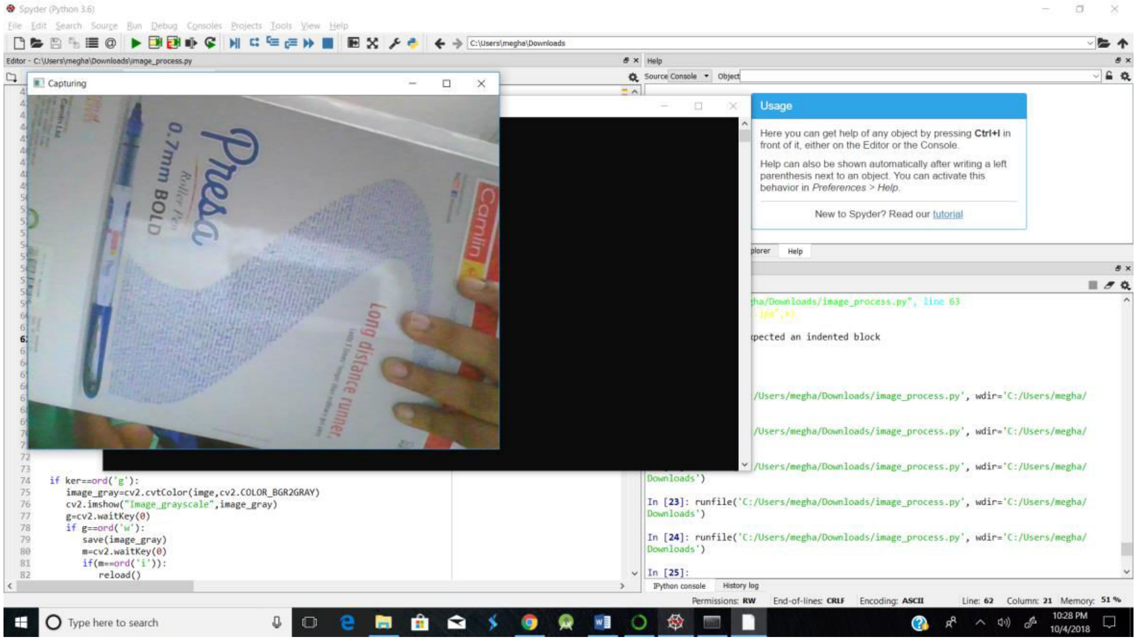
Results and Discussions

- Loading the image

  Compiling and running the program captures the image from camera.

  -Pressing key ‘i’ reloads the image.

  -Pressing key ‘w’ saves the image.



Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
if ker==ord('g'):
    image_gray=cv2.cvtColor(imge,cv2.COLOR_BGR2GRAY)
    cv2.imshow("Image_grayscale",image_gray)
    g=cv2.waitKey(0)
    if g==ord('w'):
        save(image_gray)
        m=cv2.waitKey(0)
        if(m==ord('i')):
            reload()
```

```
tha/Downloads/image_process.py", line 63

xpected an indented block

/Users/megha/Downloads/image_process.py', wdir='C:/Users/megha/

/Users/megha/Downloads/image_process.py', wdir='C:/Users/megha/

/Users/megha/Downloads/image_process.py', wdir='C:/Users/megha/
Downloads')

In [23]: runfile('C:/Users/megha/Downloads/image_process.py', wdir='C:/Users/megha/
Downloads')

In [24]: runfile('C:/Users/megha/Downloads/image_process.py', wdir='C:/Users/megha/
Downloads')

In [25]:
```

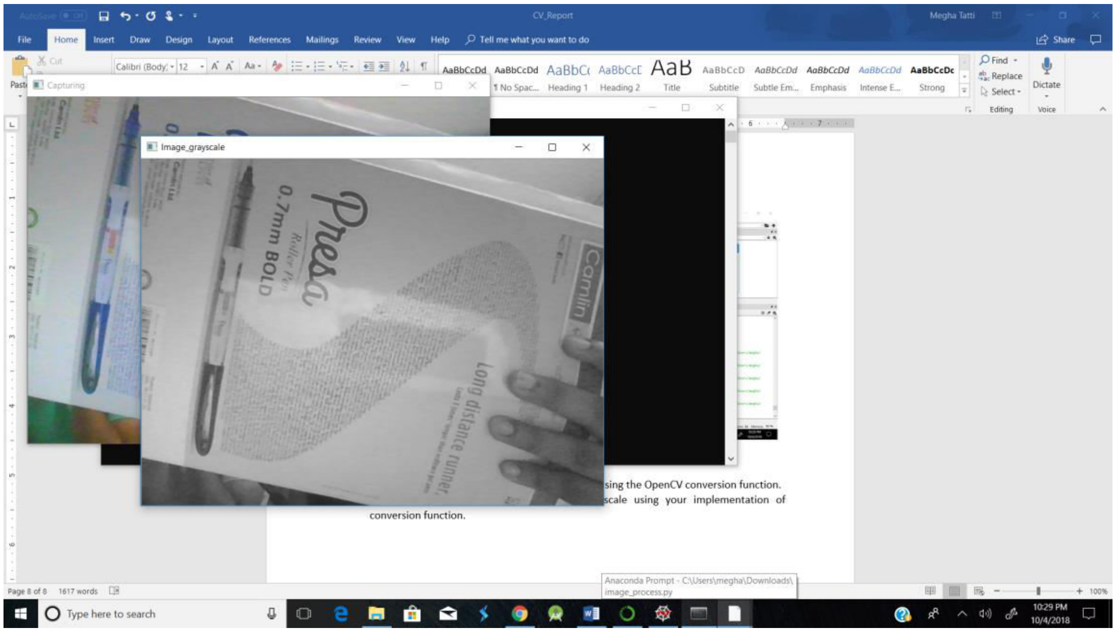
Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 62 Column: 21 Memory: 51 %

10:28 PM 10/4/2018

- Gray Scale Image

  -Pressing key ’g’ converts the image to grayscale using the OpenCV conversion function.

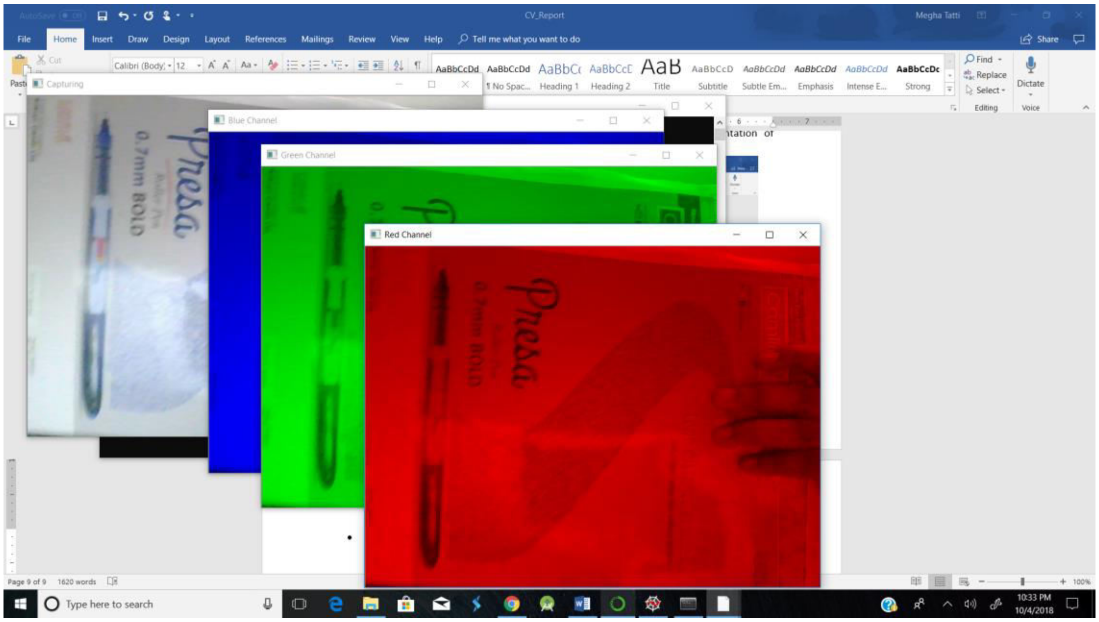
  -Pressing key ‘G’ converts the image to grayscale using your implementation of conversion function.



sing the OpenCV conversion function.
scale using your implementation of
conversion function.

- RGB Color Gradient

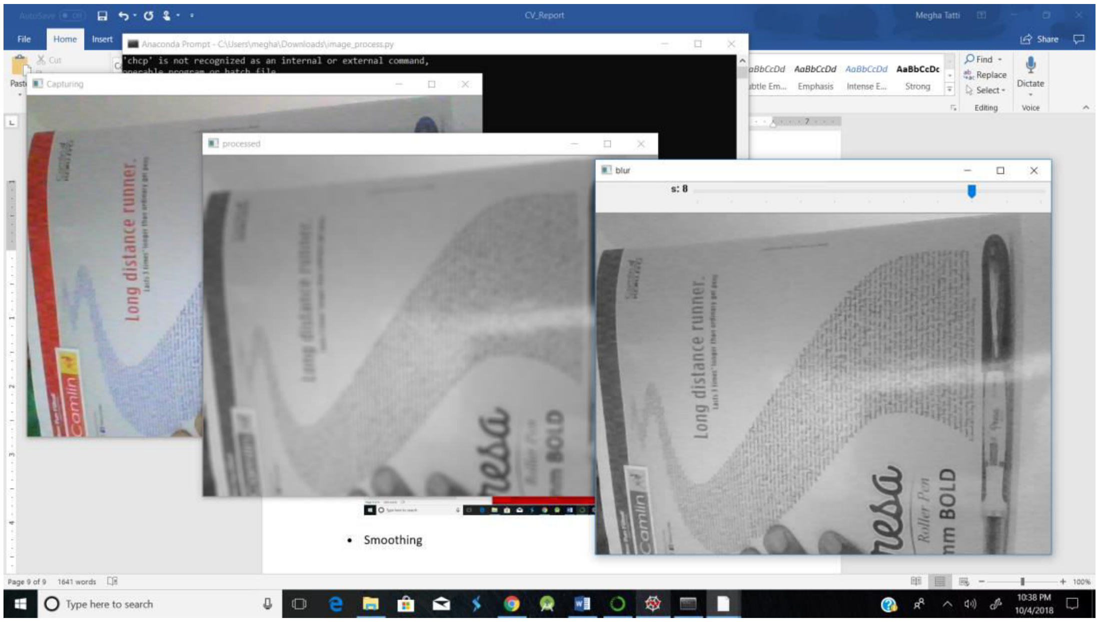
  -Pressing ‘c’ cycles through the color channels of the image showing a different channel every time the key is pressed.



- Smoothing

  -Pressing ‘s’ converts the image to grayscale and smooth it using the OpenCV function.

  -Pressing ‘S’ converts the image to grayscale and smooth it using your function which performs convolution with suitable filter.

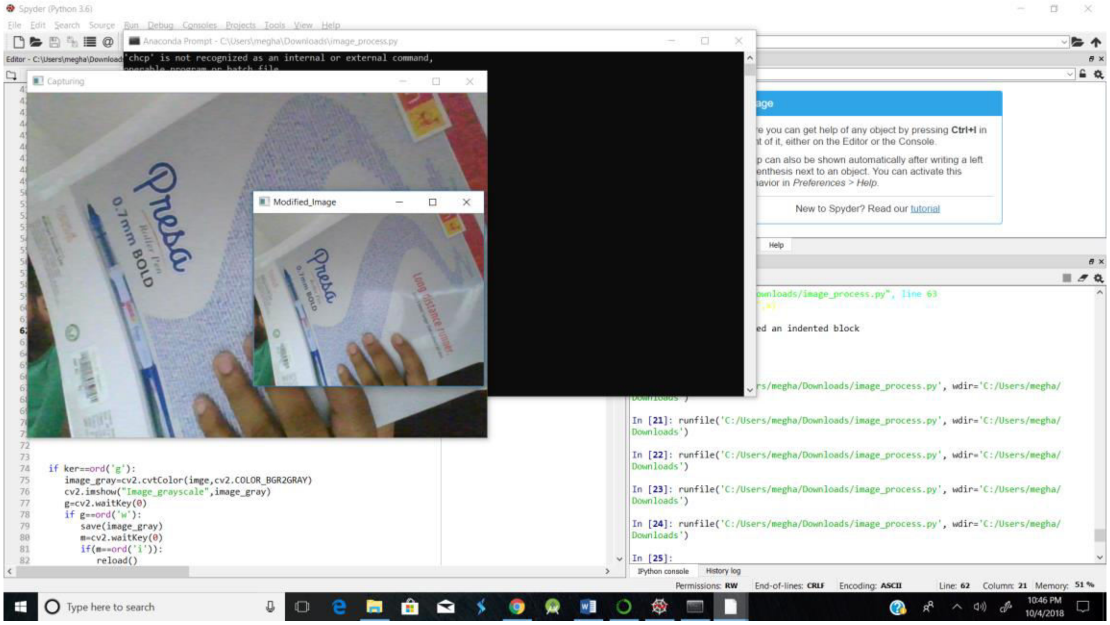


```
'chcp' is not recognized as an internal or external command,
operable program or batch file.
```

- Sampling

  -Pressing ‘d’ down samples the image by factor 2 without smoothing.

  -Pressing ‘D’ down samples the image by factor 2 with smoothing.



```
'chcp' is not recognized as an internal or external command,
operable program or batch file.
```

```
if ker==ord('g'):
    image_gray=cv2.cvtColor(imge,cv2.COLOR_BGR2GRAY)
    cv2.imshow("Image_grayscale",image_gray)
    g=cv2.waitKey(0)
    if g==ord('w'):
        save(image_gray)
        m=cv2.waitKey(0)
        if(m==ord('i')):
            reload()
```

e you can get help of any object by pressing Ctrl+I in
t of it, either on the Editor or the Console.

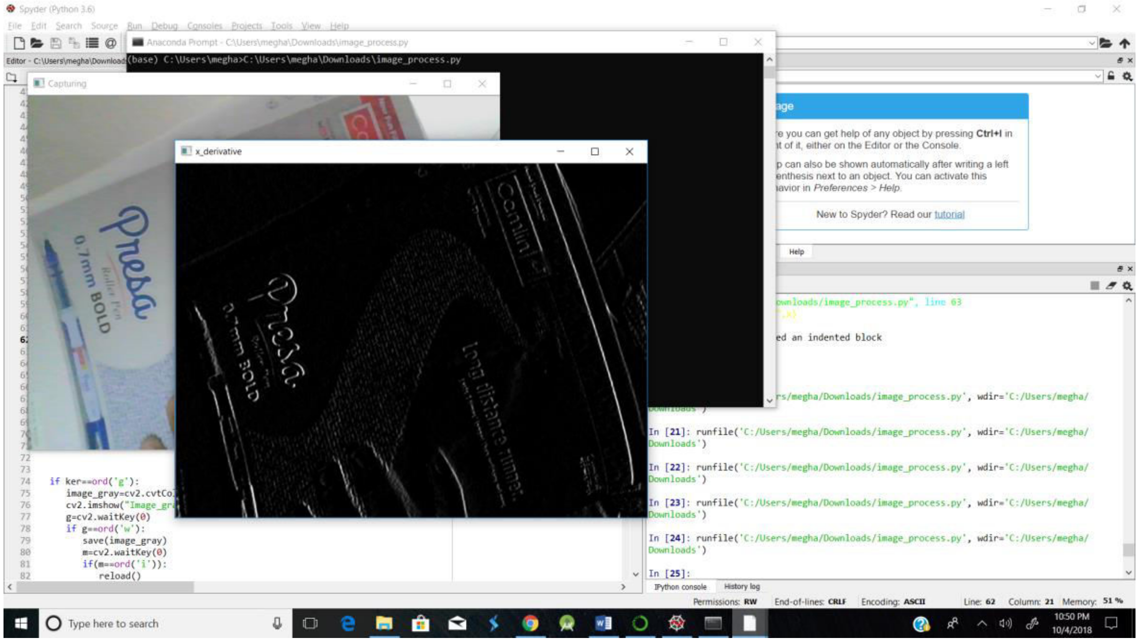
p can also be shown automatically after writing a left
enthesis next to an object. You can activate this
avior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
ownloads/image_process.py", line 63

ed an indented block

rs/megha/Downloads/image_process.py', wdir='C:/Users/megha/
Downloads')

In [21]: runfile('C:/Users/megha/Downloads/image_process.py', wdir='C:/Users/megha/
Downloads')

In [22]: runfile('C:/Users/megha/Downloads/image_process.py', wdir='C:/Users/megha/
Downloads')

In [23]: runfile('C:/Users/megha/Downloads/image_process.py', wdir='C:/Users/megha/
Downloads')

In [24]: runfile('C:/Users/megha/Downloads/image_process.py', wdir='C:/Users/megha/
Downloads')

In [25]:
```

- X-derivative

  -Pressing ‘x’ converts the image to grayscale and perform convolution with an x derivative filter.



Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Anaconda Prompt - C:\Users\megha\Downloads\image_process.py
Editor - C:\Users\megha\Download
Capturing
x_derivative
Help
IPython console
History log
Type here to search
10:50 PM
10/4/2018

```
(base) C:\Users\megha>C:\Users\megha\Downloads\image_process.py
```

e you can get help of any object by pressing **Ctrl+I** in
it of it, either on the Editor or the Console.

p can also be shown automatically after writing a left
enthesis next to an object. You can activate this
navior in *Preferences > Help*.

New to Spyder? Read our tutorial

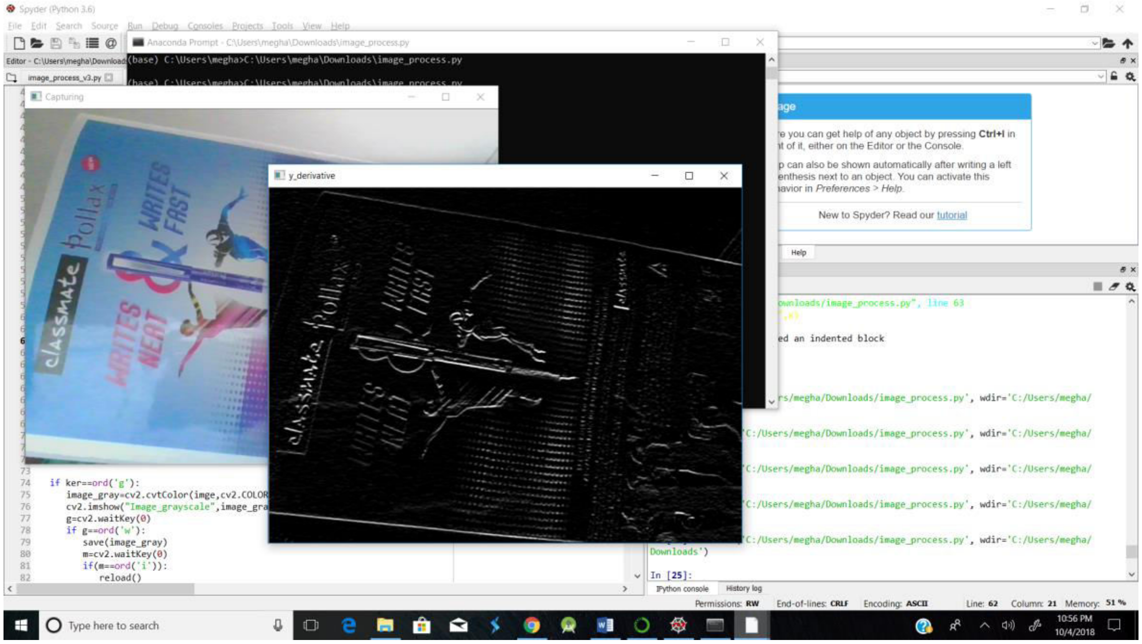
```
    if ker==ord('g'):
        image_gray=cv2.cvtCo
        cv2.imshow("Image_gr
        g=cv2.waitKey(0)
        if g==ord('w'):
            save(image_gray)
            m=cv2.waitKey(0)
            if(m==ord('i')):
                reload()
```

```
ownloads/image_process.py", line 63

ed an indented block

rs/megha/Downloads/image_process.py', wdir='C:/Users/megha/

In [21]: runfile('C:/Users/megha/Downloads/image_process.py', wdir='C:/Users/megha/
Downloads')

In [22]: runfile('C:/Users/megha/Downloads/image_process.py', wdir='C:/Users/megha/
Downloads')

In [23]: runfile('C:/Users/megha/Downloads/image_process.py', wdir='C:/Users/megha/
Downloads')

In [24]: runfile('C:/Users/megha/Downloads/image_process.py', wdir='C:/Users/megha/
Downloads')

In [25]:
```

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 62 Column: 21 Memory: 51 %

- Y-derivative

  - Pressing ‘y’ converts the image to grayscale and perform convolution with an y derivative filter.



Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Anaconda Prompt - C:\Users\megha\Downloads\image_process.py
Editor - C:\Users\megha\Download
image_process_v3.py
Capturing
y_derivative
Help
IPython console
History log
Type here to search
10:56 PM
10/4/2018

```
(base) C:\Users\megha>C:\Users\megha\Downloads\image_process.py
(base) C:\Users\megha>C:\Users\megha\Downloads\image_process.py
```

e you can get help of any object by pressing **Ctrl+I** in
it of it, either on the Editor or the Console.

p can also be shown automatically after writing a left
enthesis next to an object. You can activate this
navior in *Preferences > Help*.

New to Spyder? Read our tutorial

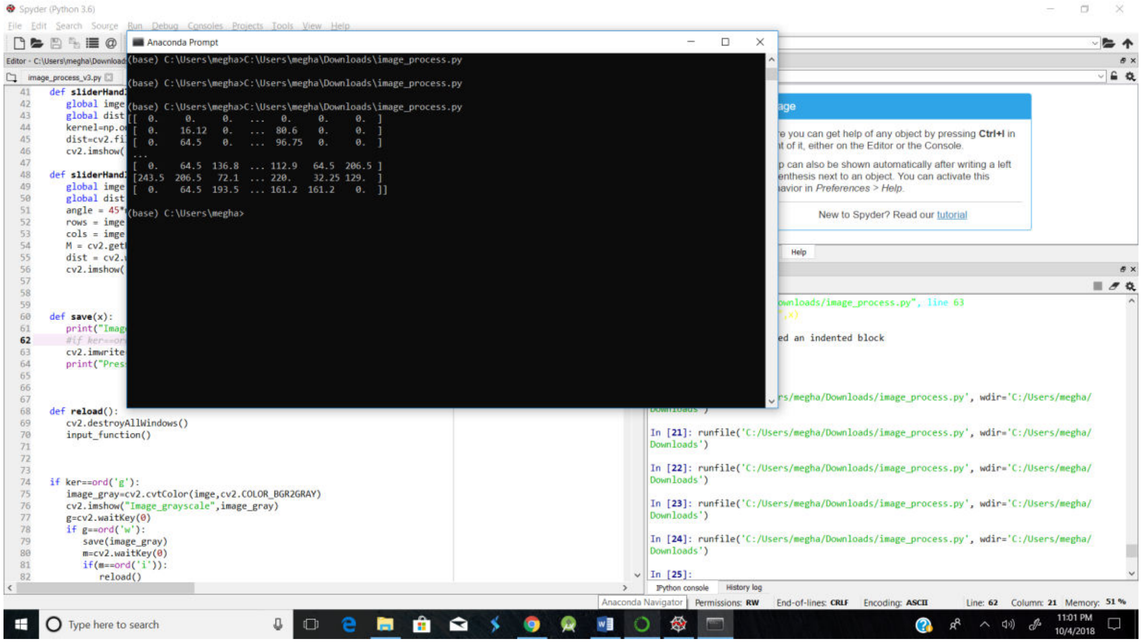
```
    if ker==ord('g'):
        image_gray=cv2.cvtColor(imge,cv2.COLOR
        cv2.imshow("Image_grayscale",image_gra
        g=cv2.waitKey(0)
        if g==ord('w'):
            save(image_gray)
            m=cv2.waitKey(0)
            if(m==ord('i')):
                reload()
```

```
ownloads/image_process.py", line 63

ed an indented block

rs/megha/Downloads/image_process.py', wdir='C:/Users/megha/

'C:/Users/megha/Downloads/image_process.py', wdir='C:/Users/megha/

'C:/Users/megha/Downloads/image_process.py', wdir='C:/Users/megha/

'C:/Users/megha/Downloads/image_process.py', wdir='C:/Users/megha/

'C:/Users/megha/Downloads/image_process.py', wdir='C:/Users/megha/
Downloads')

In [25]:
```

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 62 Column: 21 Memory: 51 %

- Magnitude

  -Pressing ‘m’ shows the magnitude of the gradient normalized to the range [0,255].



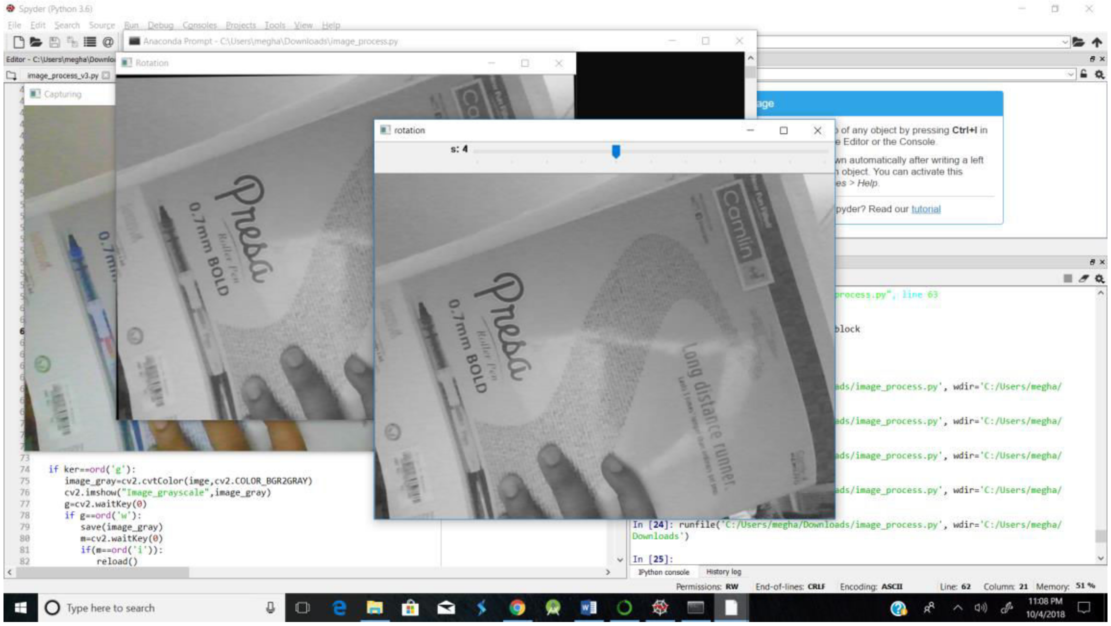
```
(base) C:\Users\megha>C:\Users\megha\Downloads\image_process.py

(base) C:\Users\megha>C:\Users\megha\Downloads\image_process.py

(base) C:\Users\megha>C:\Users\megha\Downloads\image_process.py
[[  0.      0.      0.    ...   0.      0.      0.  ]
 [  0.     16.12    0.    ...  80.6     0.      0.  ]
 [  0.     64.5     0.    ...  96.75    0.      0.  ]
 ...
 [  0.     64.5   136.8   ... 112.9    64.5   206.5 ]
 [243.5  206.5    72.1   ... 220.     32.25 129.  ]
 [  0.     64.5   193.5   ... 161.2   161.2     0.  ]]

(base) C:\Users\megha>
```

```
def sliderHand
    global imge
    global dist
    kernel=np.o
    dist=cv2.fi
    cv2.imshow(

def sliderHand
    global imge
    global dist
    angle = 45*
    rows = imge
    cols = imge
    M = cv2.get
    dist = cv2.
    cv2.imshow(


def save(x):
    print("Imag
    #if ker==or
    cv2.imwrite
    print("Pres


def reload():
    cv2.destroyAllWindows()
    input_function()


if ker==ord('g'):
    image_gray=cv2.cvtColor(imge,cv2.COLOR_BGR2GRAY)
    cv2.imshow("Image_grayscale",image_gray)
    g=cv2.waitKey(0)
    if g==ord('w'):
        save(image_gray)
        m=cv2.waitKey(0)
        if(m==ord('i')):
            reload()
```

```
ownloads/image_process.py", line 63
,x)

ed an indented block

rs/megha/Downloads/image_process.py', wdir='C:/Users/megha/
Downloads')

In [21]: runfile('C:/Users/megha/Downloads/image_process.py', wdir='C:/Users/megha/
Downloads')

In [22]: runfile('C:/Users/megha/Downloads/image_process.py', wdir='C:/Users/megha/
Downloads')

In [23]: runfile('C:/Users/megha/Downloads/image_process.py', wdir='C:/Users/megha/
Downloads')

In [24]: runfile('C:/Users/megha/Downloads/image_process.py', wdir='C:/Users/megha/
Downloads')

In [25]:
```

Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
Anaconda Prompt
Editor - C:\Users\megha\Download
image_process_v3.py
e you can get help of any object by pressing Ctrl+I in
nt of it, either on the Editor or the Console.
p can also be shown automatically after writing a left
enthesis next to an object. You can activate this
avior in Preferences > Help.
New to Spyder? Read our tutorial
Help
IPython console
History log
Anaconda Navigator
Permissions: RW
End-of-lines: CRLF
Encoding: ASCII
Line: 62
Column: 21
Memory: 51 %
Type here to search
11:01 PM
10/4/2018

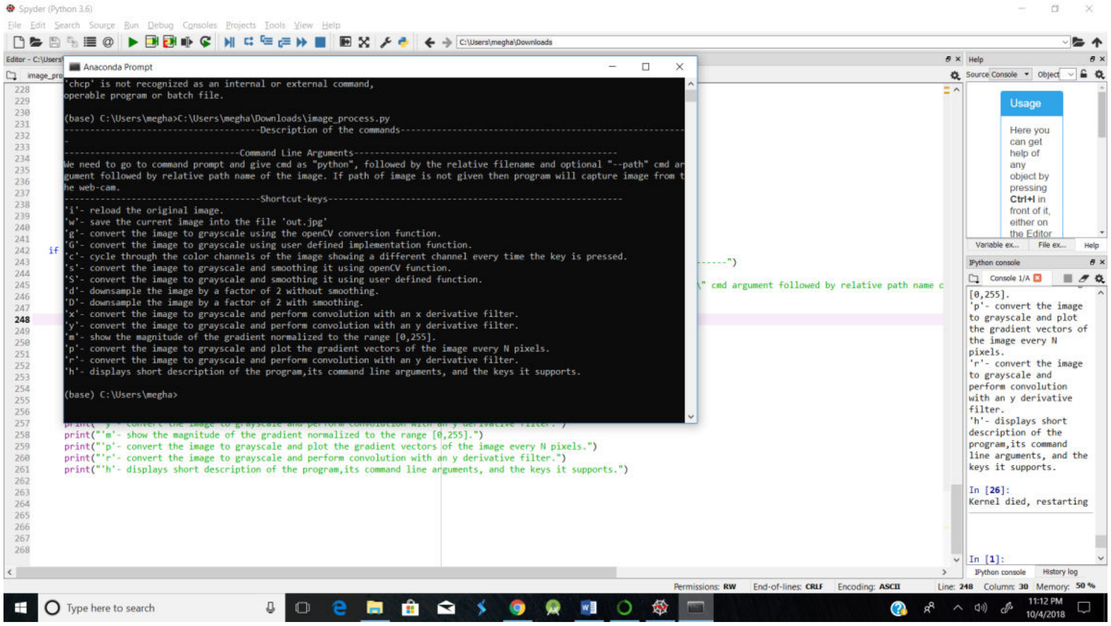
- Rotation
  -Pressing ‘r’ converts to gray scale and rotates the image.



```
if ker==ord('g'):
    image_gray=cv2.cvtColor(imge,cv2.COLOR_BGR2GRAY)
    cv2.imshow("Image_grayscale",image_gray)
    g=cv2.waitKey(0)
    if g==ord('w'):
        save(image_gray)
        m=cv2.waitKey(0)
        if(m==ord('i')):
            reload()
```

```
process.py", line 63

block

ads/image_process.py', wdir='C:/Users/megha/

ads/image_process.py', wdir='C:/Users/megha/

ads/image_process.py', wdir='C:/Users/megha/

ads/image_process.py', wdir='C:/Users/megha/

In [24]: runfile('C:/Users/megha/Downloads/image_process.py', wdir='C:/Users/megha/
Downloads')

In [25]:
```

- Short Description
  -Pressing ‘h’ gives short description of the program.


Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\megha\Downloads
Anaconda Prompt
Help
Source Console Object
Usage
Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor
Variable ex... File ex... Help
IPython console
Console 1/A
IPython console History log
Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 248 Column: 30 Memory: 50 %
Type here to search
11:12 PM
10/4/2018

```
'chcp' is not recognized as an internal or external command,
operable program or batch file.

(base) C:\Users\megha>C:\Users\megha\Downloads\image_process.py
-------------------------------------Description of the commands-----------------------------------------------------
-
--------------------------------Command Line Arguments---------------------------------------------------
We need to go to command prompt and give cmd as "python", followed by the relative filename and optional "--path" cmd ar
gument followed by relative path name of the image. If path of image is not given then program will capture image from t
he web-cam.
-------------------------------------Shortcut-keys-------------------------------------------------------
'i'- reload the original image.
'w'- save the current image into the file 'out.jpg'
'g'- convert the image to grayscale using the openCV conversion function.
'G'- convert the image to grayscale using user defined implementation function.
'c'- cycle through the color channels of the image showing a different channel every time the key is pressed.
's'- convert the image to grayscale and smoothing it using openCV function.
'S'- convert the image to grayscale and smoothing it using user defined function.
'd'- downsample the image by a factor of 2 without smoothing.
'D'- downsample the image by a factor of 2 with smoothing.
'x'- convert the image to grayscale and perform convolution with an x derivative filter.
'y'- convert the image to grayscale and perform convolution with an y derivative filter.
'm'- show the magnitude of the gradient normalized to the range [0,255].
'p'- convert the image to grayscale and plot the gradient vectors of the image every N pixels.
'r'- convert the image to grayscale and perform convolution with an y derivative filter.
'h'- displays short description of the program,its command line arguments, and the keys it supports.

(base) C:\Users\megha>
```

```
print("'m'- show the magnitude of the gradient normalized to the range [0,255].")
print("'p'- convert the image to grayscale and plot the gradient vectors of the image every N pixels.")
print("'r'- convert the image to grayscale and perform convolution with an y derivative filter.")
print("'h'- displays short description of the program,its command line arguments, and the keys it supports.")
```

```
[0,255].
'p'- convert the image
to grayscale and plot
the gradient vectors of
the image every N
pixels.
'r'- convert the image
to grayscale and
perform convolution
with an y derivative
filter.
'h'- displays short
description of the
program,its command
line arguments, and the
keys it supports.

In [26]:
Kernel died, restarting

In [1]:
```

References:

- https://stackoverflow.com/search?q=opencv
- https://docs.opencv.org/2.4/doc/tutorials/introduction/display_image/display_image.html
- https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_gui/py_image_display/py_image_display.html