3rd

```
import cv2

# Read the original image
img = cv2.imread('download.jpg')
# Display original image
cv2.imshow('Original', img)
cv2.waitKey(0)

# Convert to graycsale
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Blur the image for better edge detection
img_blur = cv2.GaussianBlur(img_gray, (3,3), 0)

# Sobel Edge Detection
# sobelx = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=0, ksize=5) # Sobel Edge Detection on the X axis
# sobely = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=0, dy=1, ksize=5) # Sobel Edge Detection on the Y axis
# sobelxy = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=1, ksize=5) # Combined X and Y Sobel Edge Detection
# # Display Sobel Edge Detection Images
# cv2.imshow('Sobel X', sobelx)
# cv2.waitKey(0)
# cv2.imshow('Sobel Y', sobely)
# cv2.waitKey(0)
# cv2.imshow('Sobel X Y using Sobel() function', sobelxy)
# cv2.waitKey(0)

# Canny Edge Detection
edges = cv2.Canny(image=img_blur, threshold1=100, threshold2=200) # Canny Edge Detection
# Display Canny Edge Detection Image
```

```
cv2.imshow('Canny Edge Detection', edges)
cv2.waitKey(0)

cv2.destroyAllWindows()
import cv2
import numpy as np

img = cv2.imread('download.jpg')
print(img.shape) # Print image shape
cv2.imshow("original", img)

# Cropping an image
cropped_image = img[80:280, 150:330]

# Display cropped image
cv2.imshow("cropped", cropped_image)

# Save the cropped image
cv2.imwrite("Cropped Image.jpg", cropped_image)

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

```
(183, 275, 3)
```

4th

```
 import cv2

# Read the original image
img = cv2.imread('download.jpg')
# Display original image
cv2.imshow('Original', img)
```

```
cv2.waitKey(0)

# Convert to graycsale
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Blur the image for better edge detection
img_blur = cv2.GaussianBlur(img_gray, (3,3), 0)

# Sobel Edge Detection
# sobelx = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=0, ksize=5) # Sobel Edge Detection on the X axis
# sobely = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=0, dy=1, ksize=5) # Sobel Edge Detection on the Y axis
# sobelxy = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=1, ksize=5) # Combined X and Y Sobel Edge Detection
# # Display Sobel Edge Detection Images
# cv2.imshow('Sobel X', sobelx)
# cv2.waitKey(0)
# cv2.imshow('Sobel Y', sobely)
# cv2.waitKey(0)
# cv2.imshow('Sobel X Y using Sobel() function', sobelxy)
# cv2.waitKey(0)

# Canny Edge Detection
edges = cv2.Canny(image=img_blur, threshold1=100, threshold2=200) # Canny Edge Detection
# Display Canny Edge Detection Image
cv2.imshow('Canny Edge Detection', edges)
cv2.waitKey(0)

cv2.destroyAllWindows()
import cv2
import numpy as np
```

```
img = cv2.imread('download.jpg')
print(img.shape) # Print image shape
cv2.imshow("original", img)

# Cropping an image
cropped_image = img[80:280, 150:330]

# Display cropped image
cv2.imshow("cropped", cropped_image)

# Save the cropped image
cv2.imwrite("Cropped Image.jpg", cropped_image)

cv2.waitKey(0)
cv2.destroyAllWindows()
output (183, 275, 3)
```

5th

```
import tkinter
from tkinter import *
from tkinter import messagebox
import PIL
from PIL import ImageTk
from PIL import Image
# from PIL import Image, ImageTK
from PIL import ImageFont
from PIL import ImageDraw
import cv2
import os
top = Toplevel()
```

```
# //IMAGE
img2 = ImageTk.PhotoImage(Image.open("download.jpg"))
image1 = cv2.imread('download1.jpg')
img = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)
panel = tkinter.Label(top, image = img2)
panel.pack(side = "bottom", fill = "both",expand = "yes")

# //LABEL
# var = StringVar()
# # label = Label( root, textvariable=var, relief=RAISED)
# var.set("Normal Thresolding")

# //BINARY
def BINARY():
   cv2.imshow('original',img)
   ret, thresh1 = cv2.threshold(img, 120, 255, cv2.THRESH_BINARY)
   cv2.imshow('BINARY', thresh1)
   if cv2.waitKey(0) & 0xff == 27:
      cv2.destroyAllWindows()
B = tkinter.Button(top, text ="BINARY", command = BINARY)
B.pack()

# //BINARY_INV
def BINARY_INV():
   cv2.imshow('original',img)
   ret, thresh2 = cv2.threshold(img, 120, 255, cv2.THRESH_BINARY)
   cv2.imshow('BINARY_INV', thresh2)
   if cv2.waitKey(0) & 0xff == 27:
      cv2.destroyAllWindows()
B = tkinter.Button(top, text ="BINARY_INV", command = BINARY_INV)
B.pack()
```

```
# //TOZERO
def TOZERO():
    cv2.imshow('original',img)
    ret, thresh3 = cv2.threshold(img, 120, 255, cv2.THRESH_TOZERO)
    cv2.imshow('TOZERO', thresh3)
    if cv2.waitKey(0) & 0xff == 27:
        cv2.destroyAllWindows()
B = tkinter.Button(top, text ="TOZERO", command = TOZERO)
B.pack()

# //TOZERO_INV
def TOZERO_INV():
    cv2.imshow('original',img)
    ret, thresh4 = cv2.threshold(img, 120, 255, cv2.THRESH_TOZERO_INV)
    cv2.imshow('TOZERO_INV', thresh4)
    if cv2.waitKey(0) & 0xff == 27:
        cv2.destroyAllWindows()
B = tkinter.Button(top, text ="TOZERO_INV", command = TOZERO_INV)
B.pack()

# //TRUNC
def TRUNC():
    cv2.imshow('original',img)
    ret, thresh5 = cv2.threshold(img, 120, 255, cv2.THRESH_TRUNC)
    cv2.imshow('TRUNC', thresh5)
    if cv2.waitKey(0) & 0xff == 27:
        cv2.destroyAllWindows()
B = tkinter.Button(top, text ="TRUNC", command = TRUNC)
B.pack()
```

```
# # //LABEL
# B = tkinter.Button(top, text ="Adaptive Thresholding")
# B.pack()

# //MEAN_C
def MEAN_C():
    cv2.imshow('original',img)
    thresh6 = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_MEAN_C,
cv2.THRESH_BINARY, 199, 5)
    cv2.imshow('MEAN_C', thresh6)
    if cv2.waitKey(0) & 0xff == 27:
        cv2.destroyAllWindows()
B = tkinter.Button(top, text ="MEAN_C", command = MEAN_C)
B.pack()

# //GAUSSIAN_C
def GAUSSIAN_C():
    cv2.imshow('original',img)
    thresh7 = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY, 199, 5)
    cv2.imshow('GAUSSIAN_C', thresh7)
    if cv2.waitKey(0) & 0xff == 27:
        cv2.destroyAllWindows()
B = tkinter.Button(top, text ="GAUSSIAN_C", command = GAUSSIAN_C)
B.pack()
top.mainloop()
```

6th

```
# importing libraries
import numpy as np
import cv2
from matplotlib import pyplot as plt
```

```
img = cv2.imread('gray.jpg',1)

dst = cv2.fastNlMeansDenoisingColored(img, None, 10, 10, 7, 15)

plt.subplot(121), plt.imshow(img)

plt.subplot(122), plt.imshow(dst)


plt.show()

import cv2

from matplotlib import pyplot as plt

img = cv2.imread('color.jpg')

avging = cv2.blur(img,(10,10))

# cv2.imshow('Averaging',avging)

plt.subplot(121), plt.imshow(img)

plt.subplot(122), plt.imshow(avging)

cv2.waitKey(0)

gausBlur = cv2.GaussianBlur(img, (5,5),0)

cv2.destroyAllWindows()

import cv2

import numpy as np

cap = cv2.VideoCapture('sample.mp4')

while (cap.isOpened()):

        ret, frame = cap.read()

        frame = cv2.resize(frame, (540, 380), fx = 0, fy = 0,

                                                interpolation = cv2.INTER_CUBIC)

        cv2.imshow('Frame', frame)

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        Thresh = cv2.adaptiveThreshold(gray, 255, cv2.ADAPTIVE_THRESH_MEAN_C,


        cv2.THRESH_BINARY_INV, 11, 2)

        cv2.imshow('Thresh', Thresh)

        if cv2.waitKey(25) & 0xFF == ord('q'):

                break
```

```
cap.release()
cv2.destroyAllWindows()
import cv2
cap = cv2.VideoCapture(0)
i = 0

while(cap.isOpened()):
    ret, frame = cap.read()
    if ret == False:
        break
    cv2.imwrite('Frame'+str(i)+'.jpg', frame)
    i += 1
cap.release()
cv2.destroyAllWindows()
```

7th

```
# import Opencv
import cv2

# import Numpy
import numpy as np

# read a image using imread
img = cv2.imread('download.jpg',0)

# creating a Histograms Equalization
# of a image using cv2.equalizeHist()
equ = cv2.equalizeHist(img)

# stacking images side-by-side
res = np.hstack((img, equ))
```

```
# show image input vs output
cv2.imshow('image', res)


cv2.waitKey(0)
cv2.destroyAllWindows()
import cv2
import numpy as np
import matplotlib.pyplot as plt
# Reading the image from the present directory
image = cv2.imread("download.jpg")
# Resizing the image for compatibility
image = cv2.resize(image, (500, 600))

# The initial processing of the image
# image = cv2.medianBlur(image, 3)
image_bw = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# The declaration of CLAHE
# clipLimit -> Threshold for contrast limiting
clahe = cv2.createCLAHE(clipLimit = 5)
final_img = clahe.apply(image_bw) + 30

# Ordinary thresholding the same image
_, ordinary_img = cv2.threshold(image_bw, 155, 255, cv2.THRESH_BINARY)

# Showing all the three images
#cv2.imshow("ordinary threshold", ordinary_img)
#cv2.imshow("CLAHE image", final_img)
plt.imshow(ordinary_img)
plt.imshow(final_img)
```