image=cv2.imread("light.jfif")

#to convert the image in greyscale

img=cv2.cvtColor(image,cv2.COLOR\_BGR2GRAY)

ret,th1=cv2.threshold(img,160,255,cv2.THRESH\_BINARY)

th2=cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE\_THRESH\_MEAN\_C,cv2.THRESH\_BINARY,11,2)

th3=cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C,cv2.THRESH\_BINARY,11,2)

plt.subplot(141)

plt.title("Original Image")

plt.xticks([]),plt.yticks([])

plt.imshow(img,cmap='gray')

plt.subplot(142)

plt.title("binary")

plt.xticks([]),plt.yticks([])

plt.imshow(th1,cmap='gray')

plt.subplot(143)

plt.title("Mean Image")

plt.xticks([]),plt.yticks([])

plt.imshow(th2,cmap='gray')

plt.subplot(144)

plt.title("Gaussian Image")

plt.xticks([]),plt.yticks([])

plt.imshow(th3,cmap='gray')

Graphical user interface, text, application

Description automatically generated

import numpy as np

import cv2

from google.colab.patches import cv2\_imshow

image1 = np.zeros((400,400), dtype = "uint8")

cv2\_imshow(image1)

Graphical user interface, text, application, chat or text message

Description automatically generated

cv2.rectangle(image1, (100,100), (250,250), 255, -1)

cv2\_imshow(image1)

A picture containing logo

Description automatically generated

image2 = np.zeros((400,400), dtype = "uint8")

cv2.circle(image2, (150, 150), 90 ,255, -1)

cv2\_imshow(image2)

A screenshot of a computer

Description automatically generated with low confidence

bitand = cv2.bitwise\_and(image1, image2)

cv2\_imshow(bitand)

A picture containing application

Description automatically generated

bitor = cv2.bitwise\_or(image1, image2)

cv2\_imshow(bitor)

A picture containing graphical user interface

Description automatically generated

bitxor = cv2.bitwise\_xor(image1, image2)

cv2\_imshow(bitxor)

Icon

Description automatically generated with medium confidence

import cv2

from google.colab.patches import cv2\_imshow

from matplotlib import pyplot as plt

img1 = cv2.imread("strawberry.jpg")

cv2\_imshow(img1)

%matplotlib inline

#computing the histogram of the blue channel of the image

hist = cv2.calcHist([img1],[0], None,[256],[0,256])

#Plot the above computed histogram

plt.plot(hist, color = "b")

plt.title("Image Histogram For Blue Channel")

plt.show()

Graphical user interface, text

Description automatically generated

A close up of a strawberry

Description automatically generated

Chart, histogram

Description automatically generated

#Define colors to plot the histograms

colors = ('b','g','r')

#%matplotlib inline

#Compute and plot the image histograms

for i, color in enumerate(colors):

  hist = cv2.calcHist([img1],[i],None,[256],[0,256])

  plt.plot(hist, color = color)

plt.title("Image Histogram")

plt.show()

Chart

Description automatically generated

%matplotlib inline

import numpy as np

import cv2

from google.colab.patches import cv2\_imshow

img2 = cv2.imread("dull1.png", 0)

cv2\_imshow(img2)

equal = cv2.equalizeHist(img2)

cv2\_imshow(equal)

Graphical user interface, website

Description automatically generated

img1 = cv2.imread("vampire.jfif")

cv2\_imshow(img1)

plt.hist(img1.ravel(), bins = 256, color = "cyan")

plt.show()

red = img1[:, :, 0]

plt.hist(red.ravel(), bins = 256, color = 'red')

plt.show()

blue = img1[:,:,2]

plt.hist(blue.ravel(), bins = 256, color = 'blue')

plt.show()

Graphical user interface, text, application

Description automatically generated

Graphical user interface

Description automatically generated

Chart

Description automatically generated

%matplotlib inline

img = cv2.imread('home.jfif', 0)

cv2\_imshow(img)

#creat a mask

mask = np.zeros(img.shape[:2], np.uint8)

mask[50:200, 50:400] = 255

masked\_img = cv2.bitwise\_and(img, img, mask = mask)

#Calculate histogram with mask and without mask

#Check third argument for mask

hist\_full = cv2.calcHist([img],[0], None, [256], [0,256])

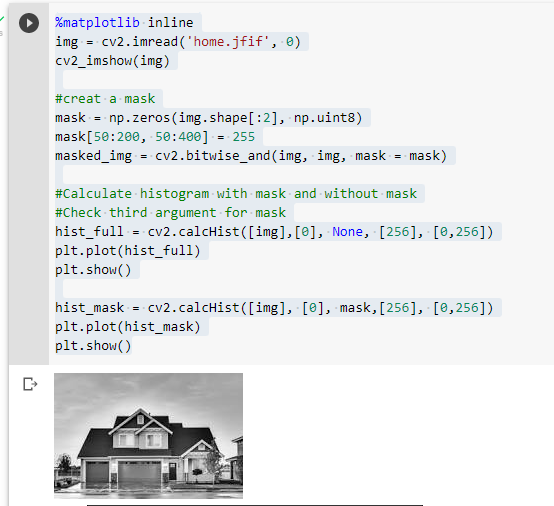
plt.plot(hist\_full)

plt.show()

hist\_mask = cv2.calcHist([img], [0], mask,[256], [0,256])

plt.plot(hist\_mask)

plt.show()



Graphical user interface, chart, histogram

Description automatically generated

import numpy as np

import cv2

from google.colab.patches import cv2\_imshow

from matplotlib import pyplot as plt

%matplotlib inline

img=cv2.imread('home.jfif',0)

cv2\_imshow(img)

#Create a Mask

mask=np.zeros(img.shape[:2],np.uint8)

mask[50:200,50:400]=255

masked\_img=cv2.bitwise\_and(img,img,mask=mask)

hist\_full=cv2.calcHist([img],[0],None,[256],[0,256])

plt.plot(hist\_full)

plt.show()

hist\_mask=cv2.calcHist([img],[0],mask,[256],[0,256])

plt.subplot(222),plt.imshow(img,'gray')

plt.subplot(223),plt.imshow(masked\_img,'gray')

plt.subplot(224),

plt.plot(hist\_mask,color="green")

plt.show()

Text

Description automatically generated

Graphical user interface, chart, histogram

Description automatically generated

Graphical user interface

Description automatically generated

#3 X 3 Filtering

import cv2

import numpy as np

from google.colab.patches import cv2\_imshow

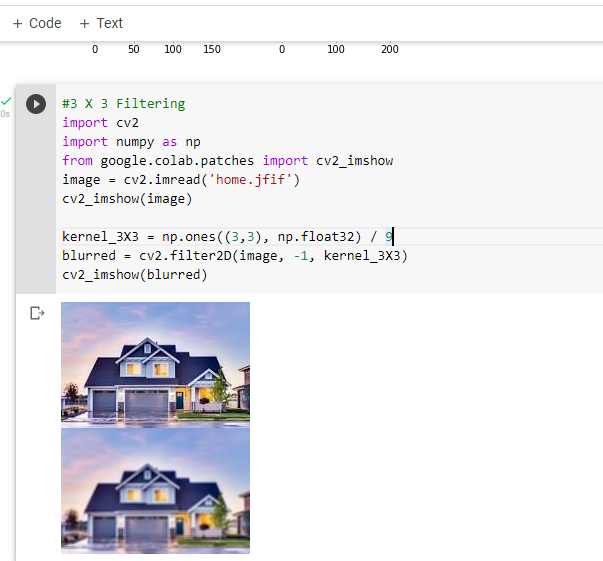
image = cv2.imread('home.jfif')

cv2\_imshow(image)

kernel\_3X3 = np.ones((3,3), np.float32) / 9

blurred = cv2.filter2D(image, -1, kernel\_3X3)

cv2\_imshow(blurred)



import cv2

import numpy as np

from google.colab.patches import cv2\_imshow

image = cv2.imread('home.jfif')

cv2\_imshow(image)

kernel\_3X3 = np.ones((5,5), np.float32) / 25

blurred = cv2.filter2D(image, -1, kernel\_3X3)

cv2\_imshow(blurred)

Graphical user interface, text, application

Description automatically generated

%matplotlib inline

import cv2

image = plt.imread('home.jfif')

medi = cv2.medianBlur(image, 5)

plt.subplot(121)

plt.title("Original Image")

plt.imshow(image)

plt.subplot(122)

plt.title("Blurred using Median")

plt.imshow(medi)

plt.show()

Graphical user interface, text, application

Description automatically generated

%matplotlib inline

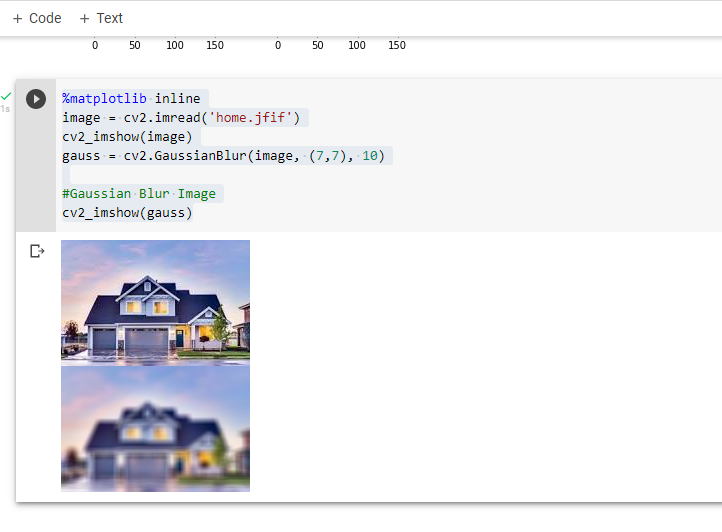
image = cv2.imread('home.jfif')

cv2\_imshow(image)

gauss = cv2.GaussianBlur(image, (7,7), 10)

#Gaussian Blur Image

cv2\_imshow(gauss)



#Sharpening Filters

import cv2

import numpy as np

from google.colab.patches import cv2\_imshow

#Reading in and diplay our image

image = cv2.imread('home.jfif')

cv2\_imshow(image)

#Create our sharpening kernel, it must equal to one eventually

kernel\_sharpening = np.array([[-1, -1, -1],

                              [-1, 9, -1],

                              [-1, -1, -1]])

#applying the sharpening kernel to the input image & displaying it.

sharpened = cv2.filter2D(image, -1, kernel\_sharpening)

cv2\_imshow(sharpened)

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

#Laplacian Filter

%matplotlib inline

img = cv2.imread('home.jfif', 0)

laplacian = cv2.Laplacian(img, cv2.CV\_64F)

sobelx = cv2.Sobel(img, cv2.CV\_64F, 1, 0, ksize = 5)

sobely = cv2.Sobel(img, cv2.CV\_64F, 0, 1, ksize = 5)

plt.subplot(2,2,1), plt.imshow(img, cmap = 'gray')

plt.title('Original'), plt.xticks([]), plt.yticks([])

plt.subplot(2,2,2), plt.imshow(img, cmap = 'gray')

plt.title('Laplacian'), plt.xticks([]), plt.yticks([])

plt.subplot(2,2,3), plt.imshow(img, cmap = 'gray')

plt.title('Sobel X'), plt.xticks([]), plt.yticks([])

plt.subplot(2,2,4), plt.imshow(img, cmap = 'gray')

plt.title('Sobel Y'), plt.xticks([]), plt.yticks([])

plt.show()

Graphical user interface, text, application

Description automatically generated

Graphical user interface, website

Description automatically generated

%matplotlib inline

img1 = cv2.imread('img1.png')

img2 = cv2.imread('img2.png')

sub = cv2.subtract(img2, img1)

cv2\_imshow(sub)

Graphical user interface, application

Description automatically generated