**1. Read an image-**

* Code-

import numpy as np

import cv2

print("OpenCV-Python Version {}".format(cv2.\_\_version\_\_))

from cv2 import imread

img = imread("/content/img1.png")

print('Datatype:',img.dtype)

print('\nDimensions', img.shape)

print(img[0,0])

**2. Display an image**

* Code-

import matplotlib.pyplot as plt

from cv2 import imread

img = imread("/content/img1.png")

plt.imshow(img)

plt.title('Displaying image using Matplotlib')

print(img[0,0])

plt.show()

**3. Color conversion:**

1. **BGR TO RGB**

* Code-

import cv2 as cv

from matplotlib import pyplot as plt

from google.colab.patches import cv2\_imshow

img = cv.imread("/content/img1.png")

from cv2 import cvtColor, COLOR\_BGR2RGB

img\_rgb = cvtColor(img,COLOR\_BGR2RGB)

plt.imshow(img\_rgb)

plt.show()

print(img\_rgb[0,0])

1. **Conversion image to greyscale**

* Code-

import cv2

import matplotlib.pyplot as plt

img = cv2.imread('/content/img1.png')

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

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

plt.show()

1. **BGR TO HSV**

* Code-

import cv2

import matplotlib.pyplot as plt

img = cv2.imread('/img1.png')

from cv2 import cvtColor, COLOR\_BGR2HSV

img\_hsv = cvtColor(img,COLOR\_BGR2HSV)

plt.imshow(img\_hsv)

plt.show()

print(img\_hsv[0,0])

**1. Splitting image in three channels:**

1. **Splitting RGB image into 3 channels**

* Code-

import cv2

import matplotlib.pyplot as plt

img\_rgb = cv2.imread('/content/img1.png')

r,g,b = cv2.split(img\_rgb)

plt.figure(figsize=(10,10)) #this is used to fix the image size

plt.subplot(221)

plt.title('Original')

plt.imshow(img\_rgb)

plt.subplot(222)

plt.title('Blue Channel')

plt.imshow(b)

plt.subplot(223)

plt.title('Green Channel')

plt.imshow(g)

plt.subplot(224)

plt.title('Red Channel')

plt.imshow(r)

1. **SPLITTING HSV IMAGE INTO 3 CHANNELS**

* Code-

import cv2

import matplotlib.pyplot as plt

img = cv2.imread('/content/img1.png')

img\_hsv = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV)

h,s,v = cv2.split(img\_hsv)

plt.figure(figsize=(10,10)) #this is used to fix the image size

plt.subplot(221)

plt.title('Original')

plt.imshow(img)

plt.subplot(222)

plt.title('Hue Channel')

plt.imshow(h)

plt.subplot(223)

plt.title('Saturation Channel')

plt.imshow(s)

plt.subplot(224)

plt.title('Value Channel')

plt.imshow(v)

1. **SPLITTING LAB IMAGE INTO 3 CHANNELS**

* Code-

import cv2

import matplotlib.pyplot as plt

img = cv2.imread('/content/img1.png')

img\_lab = cv2.cvtColor(img, cv2.COLOR\_BGR2LAB)

l,a,b = cv2.split(img\_lab)

plt.figure(figsize=(10,10)) #this is used to fix the image size

plt.subplot(221)

plt.title('Original')

plt.imshow(img)

plt.subplot(222)

plt.title('Lightness Channel')

plt.imshow(l)

plt.subplot(223)

plt.title('Green-Red Channel')

plt.imshow(a)

plt.subplot(224)

plt.title('Blue-Yellow Channel')

plt.imshow(b)

**1. Image Transformation**

1. **Translation**

* Code-

import numpy as np

import cv2 as cv

from google.colab.patches import cv2\_imshow

img = cv.imread('/content/img1.png',0)

rows , cols = img.shape

M = np.float32([[1,0,100],[0,1,100]])

dst = cv.warpAffine (img, M , (cols,rows))

cv2\_imshow(dst)

cv2.waitKey(0)

cv2.destroyAllWindows

We can see that the image is cropped, to get the full image, we can use the following code-

* Code-

import numpy as np

import cv2 as cv

from google.colab.patches import cv2\_imshow

img = cv.imread('/content/img1.png',0)

rows , cols = img.shape

M = np.float32([[1,0,100],[0,1,100]])

dst = cv.warpAffine (img, M , (cols+100,rows+100))

cv2\_imshow(dst)

cv2.waitKey(0)

cv2.destroyAllWindows

1. **Rotation along X,Y axis**

* Code-

import cv2

from google.colab import files

import numpy as np

from matplotlib import pyplot as plt

uploaded = files.upload()

file\_name = next(iter(uploaded))

img = cv2.imread(file\_name)

rotated\_x = cv2.flip(img, 0)

rotated\_y = cv2.flip(img, 1)

plt.subplot(1, 3, 1)

plt.title('Original Image')

plt.imshow(cv2.cvtColor(img, cv2.COLOR\_BGR2RGB))

plt.subplot(1, 3, 2)

plt.title('Rotated along X axis')

plt.imshow(cv2.cvtColor(rotated\_x, cv2.COLOR\_BGR2RGB))

plt.subplot(1, 3, 3)

plt.title('Rotated along Y axis')

plt.imshow(cv2.cvtColor(rotated\_y, cv2.COLOR\_BGR2RGB))

plt.show()

1. **Rotation at a particular angle**

* Code-

import cv2

from google.colab import files

import numpy as np

from matplotlib import pyplot as plt

uploaded = files.upload()

file\_name = next(iter(uploaded))

img = cv2.imread(file\_name)

angle = 45

height, width = img.shape[:2]

rotation\_matrix = cv2.getRotationMatrix2D((width/2, height/2), angle, 1)

rotated\_img = cv2.warpAffine(img, rotation\_matrix, (width, height))

plt.subplot(1, 2, 1)

plt.title('Original Image')

plt.imshow(cv2.cvtColor(img, cv2.COLOR\_BGR2RGB))

plt.subplot(1, 2, 2)

plt.title('Rotated Image')

plt.imshow(cv2.cvtColor(rotated\_img, cv2.COLOR\_BGR2RGB))

plt.show()

1. **Cropping**

* Code-

import cv2

from google.colab import files

import numpy as np

from matplotlib import pyplot as plt

uploaded = files.upload()

file\_name = next(iter(uploaded))

img = cv2.imread(file\_name)

x, y, w, h = 800, 500, 500, 400

cropped\_img = img[y:y+h, x:x+w]

plt.subplot(1, 2, 1)

plt.title('Original Image')

plt.imshow(cv2.cvtColor(img, cv2.COLOR\_BGR2RGB))

plt.subplot(1, 2, 2)

plt.title('Cropped Image')

plt.imshow(cv2.cvtColor(cropped\_img, cv2.COLOR\_BGR2RGB))

plt.show()

**1. Image Transformation**

1. **Scaling by shrinking and enlarging an image**

* Code-

import cv2

from google.colab import files

from google.colab.patches import cv2\_imshow

uploaded = files.upload()

file\_name = next(iter(uploaded))

img = cv2.imread(file\_name)

shrink\_scale\_percent = 10

enlarge\_scale\_percent = 120

shrink\_width = int(img.shape[1] \* shrink\_scale\_percent / 100)

shrink\_height = int(img.shape[0] \* shrink\_scale\_percent / 100)

enlarge\_width = int(img.shape[1] \* enlarge\_scale\_percent / 100)

enlarge\_height = int(img.shape[0] \* enlarge\_scale\_percent / 100)

shrinked\_img = cv2.resize(img, (shrink\_width, shrink\_height), interpolation=cv2.INTER\_AREA)

enlarged\_img = cv2.resize(img, (enlarge\_width, enlarge\_height), interpolation=cv2.INTER\_CUBIC)

cv2\_imshow(shrinked\_img)

cv2\_imshow(enlarged\_img)

1. **Shearing along X and Y Axis**

* Code-

import cv2

from google.colab import files

import numpy as np

from matplotlib import pyplot as plt

uploaded = files.upload()

file\_name = next(iter(uploaded))

img = cv2.imread(file\_name)

shear\_factor\_x = 0.2

shear\_factor\_y = 0.1

height, width = img.shape[:2]

M\_shear\_x = np.float32([[1, shear\_factor\_x, 0],

[0, 1, 0]])

M\_shear\_y = np.float32([[1, 0, 0],

[shear\_factor\_y, 1, 0]])

sheared\_img\_x = cv2.warpAffine(img, M\_shear\_x, (width, height))

sheared\_img\_y = cv2.warpAffine(img, M\_shear\_y, (width, height))

plt.figure(figsize=(10, 5))

plt.subplot(1, 3, 1)

plt.title('Original Image')

plt.imshow(cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

plt.subplot(1, 3, 2)

plt.title('Sheared along X axis')

plt.imshow(cv2.cvtColor(sheared\_img\_x, cv2.COLOR\_BGR2RGB))

plt.subplot(1, 3, 3)

plt.title('Sheared along Y axis')

plt.imshow(cv2.cvtColor(sheared\_img\_y, cv2.COLOR\_BGR2RGB))\

plt.show()

**1. Linear Image Filtering**

* Code-

import numpy as np

import cv2

import matplotlib.pyplot as plt

def point\_operation(img,k,l):

img = np.asarray(img,dtype = float)

img = img\*k + l

img[img>255] = 255

img[img<0] = 0

return np.asarray(img,dtype=int)

def main():

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

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

out1 = point\_operation(gray,0.5,0)

out2 = point\_operation(gray,1,10)

out3 = point\_operation(gray,0.7,25)

res = np.hstack([gray,out1,out2,out3])

plt.imshow(res, cmap = "gray")

plt.axis("off")

main()

**2. 2D Linear Image Filtering**

1. **Using Custom Kernel**

* Code-

import numpy as np

import cv2

from matplotlib import pyplot as plt

image = cv2.imread("download.jpg")

kernel1 = np.ones((5,5),np.float64)/25

img = cv2.filter2D(src = image, ddepth=-1,kernel=kernel1)

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

plt.imshow(cv2.cvtColor(image, cv2.COLOR\_BGR2RGB))

plt.title('Original Image')

plt.axis('off')

plt.subplot(1, 2, 2)

plt.imshow(cv2.cvtColor(img, cv2.COLOR\_BGR2RGB))

plt.title('Filtered Image')

plt.axis('off')

plt.show()

1. **Using cv2.blur**

* Code-

import numpy as np

import cv2

def plot\_cv\_img(input\_image,output\_image):

fig,ax = plt.subplots(nrows=1,ncols=2)

ax[0].imshow(cv2.cvtColor(input\_image,cv2.COLOR\_BGR2RGB))

ax[0].set\_title('Input Image')

ax[0].axis('off')

ax[1].imshow(cv2.cvtColor(output\_image,cv2.COLOR\_BGR2RGB))

ax[1].set\_title('Box Filter 5x5')

ax[1].axis('off')

plt.show()

def main():

img =cv2.imread("download.jpg")

Kernel\_size = (5,5)

blur = cv2.blur(img,Kernel\_size)

plot\_cv\_img(img,blur)

main()

**1. Log Transformation**

* Code-

import cv2

import numpy as np

image = cv2.imread('download.jpg', cv2.IMREAD\_GRAYSCALE)

c = 255 / np.log(1 + np.max(image))

log\_transformed = c \* np.log(1 + image)

log\_transformed = np.uint8(log\_transformed)

cv2.imshow('Original Image', image)

cv2.imshow('Log Transformed Image', log\_transformed)

cv2.waitKey(0)

cv2.destroyAllWindows()

**2. Inverse Log Transformation**

* Code-

import cv2

import numpy as np

from google.colab import files

from google.colab.patches import cv2\_imshow

uploaded = files.upload()

file\_name = next(iter(uploaded))

image = cv2.imdecode(np.frombuffer(uploaded[file\_name], np.uint8), cv2.IMREAD\_GRAYSCALE)

print("Image shape:", image.shape)

max\_intensity = np.max(image)

c = 255 / np.log(1 + max\_intensity)

inverse\_log\_transformed = np.exp(image / c) - 1

inverse\_log\_transformed = np.uint8(inverse\_log\_transformed)

cv2\_imshow(image)

cv2\_imshow(inverse\_log\_transformed)

**3. Power law Transformation**

* Code-

import cv2

import numpy as np

from google.colab import files

from google.colab.patches import cv2\_imshow

def power\_law\_transform(image, gamma):

image\_normalized = image.astype('float32') / 255.0

transformed\_image = np.power(image\_normalized, gamma)

transformed\_image = np.uint8(transformed\_image \* 255)

return transformed\_image

uploaded = files.upload()

file\_name = next(iter(uploaded))

image = cv2.imdecode(np.frombuffer(uploaded[file\_name], np.uint8), cv2.IMREAD\_GRAYSCALE)

gamma = 0.5

transformed\_image = power\_law\_transform(image, gamma)

cv2\_imshow(image)

cv2\_imshow(transformed\_image)

**1. Robert Operator**

* Code-

import cv2

import numpy as np

from matplotlib import pyplot as plt

image = cv2.imread("download.jpg", cv2.IMREAD\_GRAYSCALE)

roberts\_x = np.array([[1, 0],

[0, -1]])

roberts\_y = np.array([[0, 1],

[-1, 0]])

gradient\_x = cv2.filter2D(image, -1, roberts\_x)

gradient\_y = cv2.filter2D(image, -1, roberts\_y)

gradient\_magnitude = np.sqrt(gradient\_x\*2 + gradient\_y\*2)

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

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

plt.title('Original Image')

plt.axis('off')

plt.subplot(1, 2, 2)

plt.imshow(gradient\_magnitude, cmap='gray')

plt.title('Edge Detection (Roberts Operator)')

plt.axis('off')

plt.show()

**2. Sobel Operator**

* Code-

import cv2

import numpy as np

from matplotlib import pyplot as plt

image = cv2.imread("download.jpg", cv2.IMREAD\_GRAYSCALE)

gradient\_x = cv2.Sobel(image, cv2.CV\_64F, 1, 0, ksize=3)

gradient\_y = cv2.Sobel(image, cv2.CV\_64F, 0, 1, ksize=3)

gradient\_magnitude = np.sqrt(gradient\_x\*2 + gradient\_y\*2)

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

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

plt.title('Original Image')

plt.axis('off')

plt.subplot(1, 2, 2)

plt.imshow(gradient\_magnitude, cmap='gray')

plt.title('Edge Detection (Sobel Operator)')

plt.axis('off')

plt.show()

**3. Canny Edge Detection**

* Code-

mport cv2

import numpy as np

from matplotlib import pyplot as plt

image = cv2.imread("download.jpg", cv2.IMREAD\_GRAYSCALE)

edges = cv2.Canny(image, 100, 200) # You can adjust the thresholds here

plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)

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

plt.title('Original Image')

plt.axis('off')

plt.subplot(1, 2, 2)

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

plt.title('Canny Edge Detection')

plt.axis('off')

plt.show()

**1. Harris Corner Detection**

* Code-

import cv2

from google.colab import files

from matplotlib import pyplot as plt

uploaded = files.upload()

for filename in uploaded.keys():

img = cv2.imread(filename)

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

dst = cv2.cornerHarris(gray, 2, 3, 0.04)

dst = cv2.dilate(dst, None)

img[dst > 0.01 \* dst.max()] = [0, 0, 255]

plt.imshow(cv2.cvtColor(img, cv2.COLOR\_BGR2RGB))

plt.axis('off')

plt.show()

**1. Line Detection using Hough transformation**

* Code-

import cv2

import numpy as np

from google.colab import files

uploaded = files.upload()

for filename in uploaded.keys():

img = cv2.imread(filename)

edges = cv2.Canny(img, 50, 100)

lines = cv2.HoughLines(edges, 1, np.pi/180, 200)

if lines is not None:

for r\_thetha in lines:

arr = np.array(r\_thetha[0], dtype=np.float64)

r, thetha = arr

a = np.cos(thetha)

b = np.sin(thetha)

x0 = a \* r

y0 = b \* r

x1 = int(x0 + 1000 \* (-b))

y1 = int(y0 + 1000 \* (a))

x2 = int(x0 - 1000 \* (-b))

y2 = int(y0 - 1000 \* (a))

cv2.line(img, (x1, y1), (x2, y2), (0, 255, 0), 2)

cv2.imwrite('/content/linesdetected\_img.jpg', img)

plt.imshow(img)

plt.axis('off')

plt.show()