Experiment Title: 5

Student Name: Pratibha **UID:18BCS6093**

Branch:AIML-1 **Section/Group:B**

Semester:5 **Date of Performance:**

Subject Name:DIP LAB **Subject Code: CSF-336**

1. Aim/Overview of the practical:

Write the python program which help us to understand

a) The Color spacing and data analytics of images without using opencv .

b) Image operations and analysis using opencv

2. The task to be done:

a) The Color spacing and data analytics of images without using opencv .

b) Image operations and analysis using opencv

3. Required libraries or software

```
import numpy as np
import imageio
import matplotlib.pyplot as plt
import cv2 as cv
```

4. Algorithm/Flowchart :

5. Theme/Interests definition(For creative domains):

6. Steps for experiment/practical:

```
# ## Reading and Displaying of image with ImageIO
# In[1]:
```

```
# Importing required modules
import imageio
import numpy as np
import matplotlib.pyplot as plt
# The output of plotting commands is displayed inline within frontends.
get_ipython().run_line_magic('matplotlib', 'inline')
img = imageio.imread('aa.jpg')
plt.figure(figsize = (9,6))
plt.imshow(img)

# ## Printing the description of image (height,width,rgb value etc)
# In[2]:
print('Type of the image : ' , type(img))
# The shape of the ndarray shows that it is a three-layered matrix
print('Shape of the image : {}'.format(img.shape))
print('Image Hight {}'.format(img.shape[0]))
print('Image Width {}'.format(img.shape[1]))
print('Dimension of Image {}'.format(img.ndim))
# ndim() function return the number of dimensions of an array
print('Image size {}'.format(img.size))
print('Maximum RGB value in this image {}'.format(img.max()))
print('Minimum RGB value in this image {}'.format(img.min()))
# A specific pixel located at Row : 110 ; Column : 60
# Each channel's value of it, gradually R , G , B
print('Value of only R channel {}'.format(img[ 110, 60, 0]))
print('Value of only G channel {}'.format(img[ 110, 60, 1]))
print('Value of only B channel {}'.format(img[ 110, 60, 2]))


# ## To see the specific channel of image

# In[3]:
# To see the R channel
```

```
plt.title('R channel')
plt.ylabel('Height {}'.format(img.shape[0]))
plt.xlabel('Width {}'.format(img.shape[1]))
plt.imshow(img[ : , : , 0])
plt.show()

# In[4]:
# To see the  G channel
plt.title('G channel')
plt.ylabel('Height {}'.format(img.shape[0]))
plt.xlabel('Width {}'.format(img.shape[1]))
plt.imshow(img[ : , : , 1])
plt.show()

# In[5]:
# To see the B channel
plt.title('B channel')
plt.ylabel('Height {}'.format(img.shape[0]))
plt.xlabel('Width {}'.format(img.shape[1]))
plt.imshow(img[ : , : , 2])
plt.show()

# ## To see the intensity of channel in particular pixel range
# As an example,
# R channel: Row — 100 to 110
# G channel: Row — 200 to 210
# B channel: Row — 300 to 310

# In[6]:
# full intensity to those pixel's R channel
img[50:100 , : , 0] = 255
plt.figure( figsize = (9,6))
plt.imshow(img)
```

```
plt.show()

# In[7]:

# full intensity to those pixel's G channel
img[150:200 , : , 1] = 255
plt.figure( figsize = (10,6))
plt.imshow(img)
plt.show()

# In[8]:

# full intensity to those pixel's B channel
img[250:300 , : , 2] = 255
plt.figure( figsize = (9,6))
plt.imshow(img)
plt.show()
# In[9]:


img[ 50:250 , 300:400 , [0,1,2] ] = 100
plt.figure( figsize = (5,5))
plt.imshow(img)
plt.show()


# In[10]:
import numpy as np
pic = imageio.imread('aa.jpg')
fig, ax = plt.subplots(nrows = 1, ncols=3, figsize=(15,5))
for c, ax in zip(range(3), ax):
    # create zero matrix
    split_img = np.zeros(pic.shape, dtype="uint8")
```

```
    # 'dtype' by default: 'numpy.float64'
    # assing each channel
    split_img[ :, :, c] = pic[ :, :, c]
    # display each channel
    ax.imshow(split_img)

# In[11]:

pic = imageio.imread('aa.jpg')
plt.figure(figsize=(9,5))
plt.imshow(pic)
plt.show()



# In[12]:

low_pixel = pic < 20
# to ensure of it let's check if all values in low_pixel are True or not
if low_pixel.any() == True:
    print(low_pixel.shape)
# In[13]:
print(pic.shape)
print(low_pixel.shape)



# ## Convert Images into different image form
# In[ ]:

import cv2
import numpy as np
img = imageio.imread('aa.jpg')
retval, threshold = cv2.threshold(img, 12, 255, cv2.THRESH_BINARY)
cv2.imshow('original',img)
```

```
cv2.imshow('threshold',threshold)
cv2.waitKey(0)
cv2.destroyAllWindows()
# In[ ]:
import cv2
import numpy as np
grayscaled = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
retval, threshold = cv2.threshold(grayscaled, 10, 255, cv2.THRESH_BINARY)
cv2.imshow('original',img)
cv2.imshow('threshold',threshold)
cv2.waitKey(0)
cv2.destroyAllWindows()

# In[ ]:
import cv2
import numpy as np
img = cv2.imread('aa.jpg')
grayscaled = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
gaus = cv2.adaptiveThreshold(grayscaled, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
cv2.THRESH_BINARY, 115, 1)
cv2.imshow('original',img)
cv2.imshow('threshold',threshold)
cv2.imshow('Adaptive threshold',gaus)
#cv2.imshow('threshold2',threshold2)
cv2.waitKey(0)
cv2.destroyAllWindows()



r e t v a l 2 , t h r e s h o l d 2                                                    =
cv2.threshold(grayscaled,125,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)
cv2.imshow('original',img)
cv2.imshow('Otsu threshold',threshold2)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

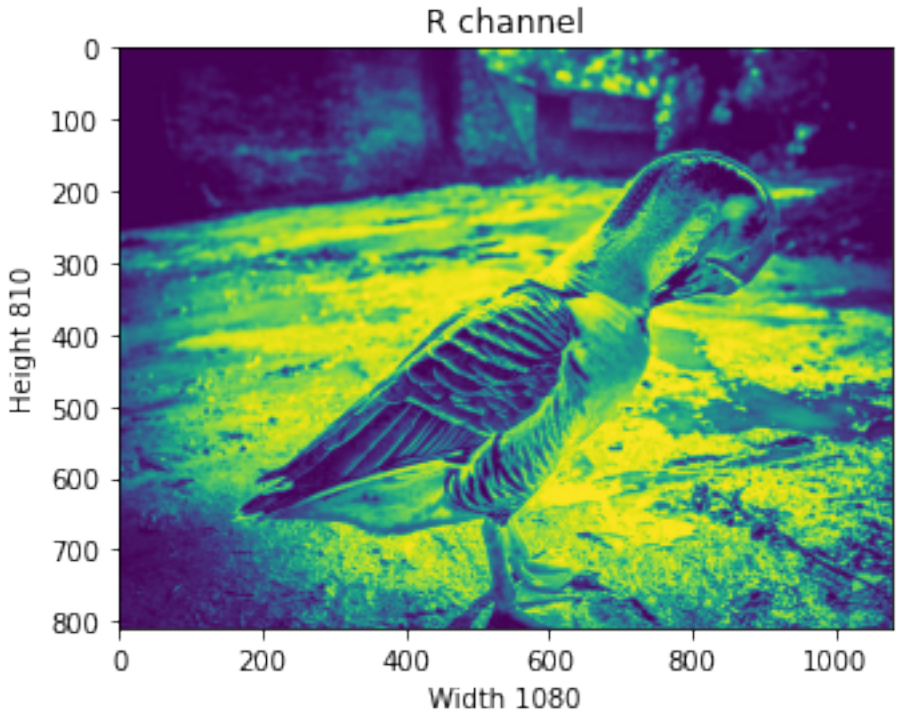
Thank You for visiting my worksheet

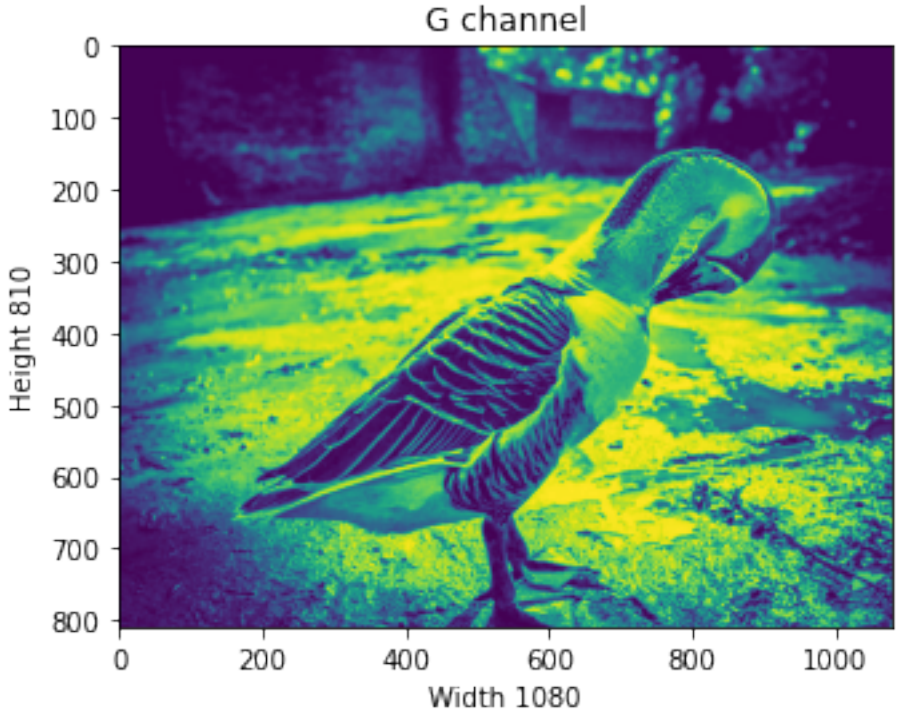
7. Observations/Discussions(For applied/experimental sciences/materials based labs):

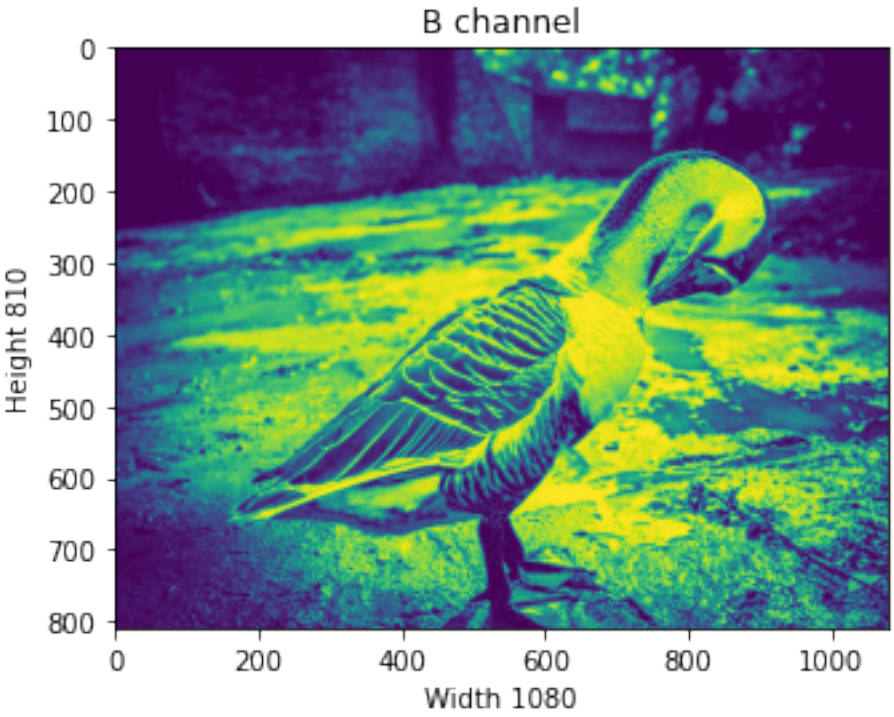
8. Percentage error (if any or applicable):

9. The command that we have learned today in the program :

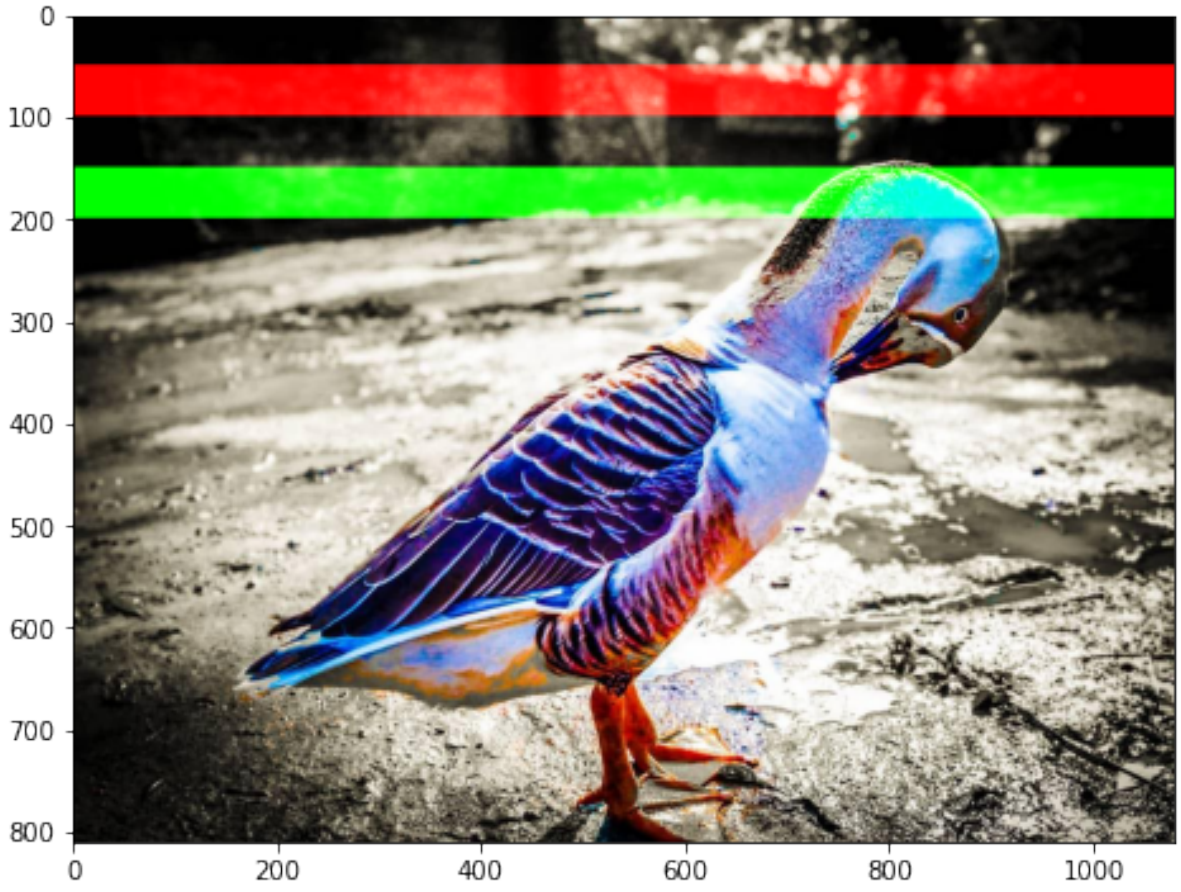
10. Result/Output/Writing Summary of the concept behind the experiment:

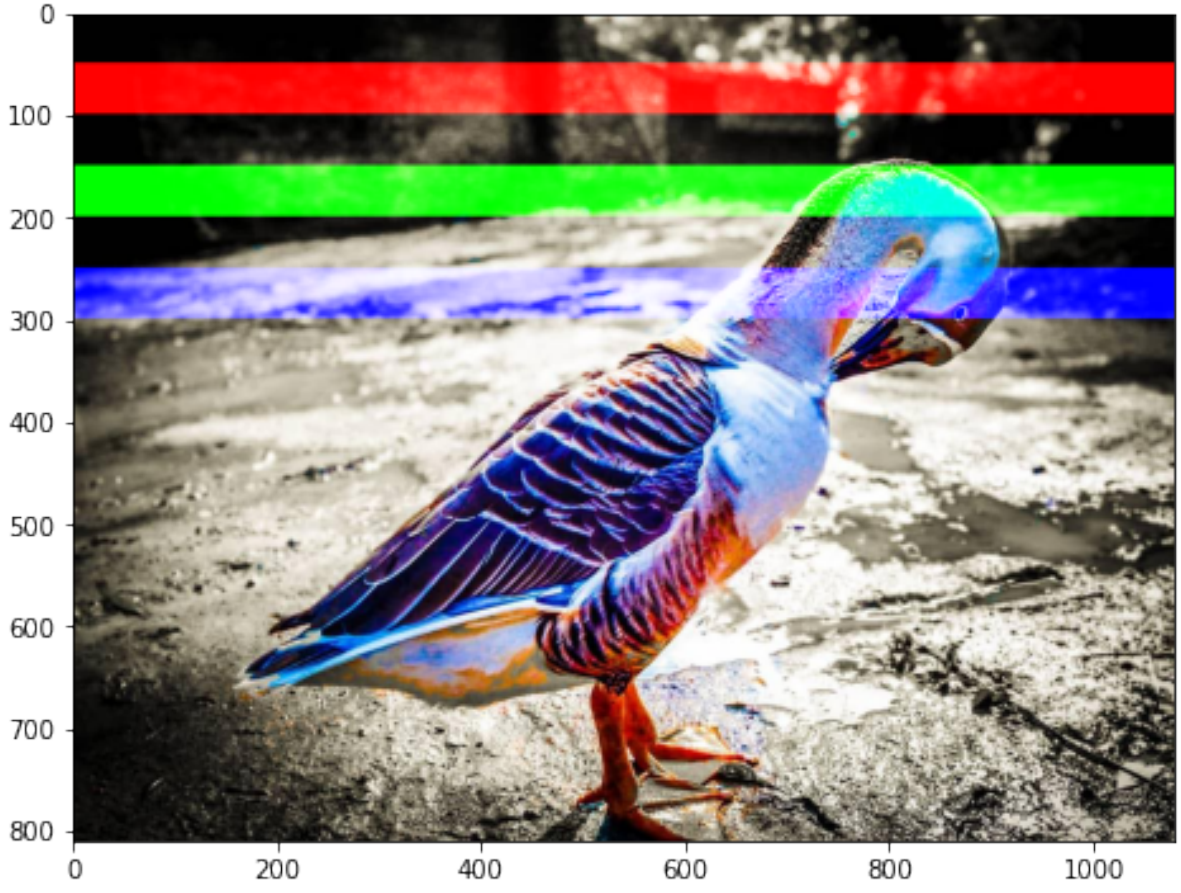

R channel
0
100
200
300
400
500
600
700
800
Height 810
0
200
400
600
800
1000
Width 1080

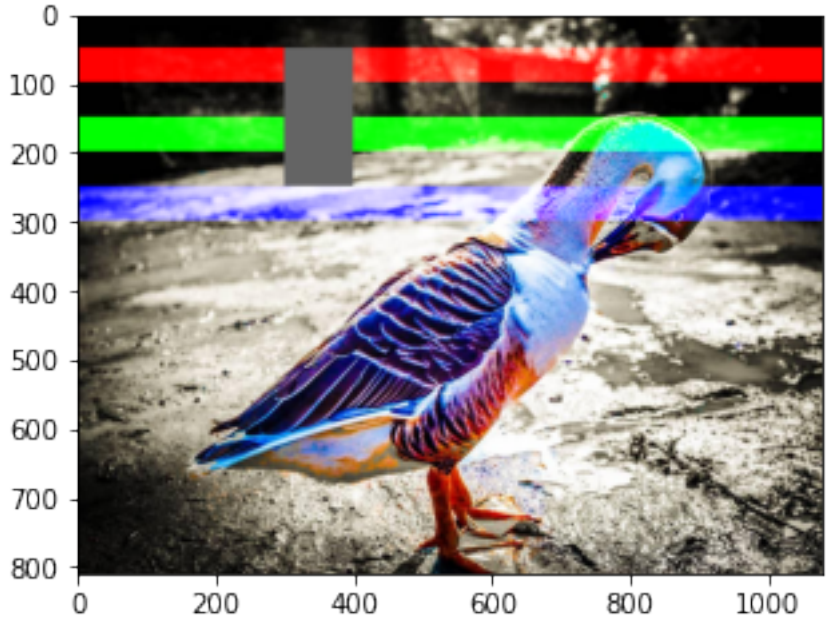

G channel
0
100
200
300
400
500
600
700
800
Height 810
0
200
400
600
800
1000
Width 1080

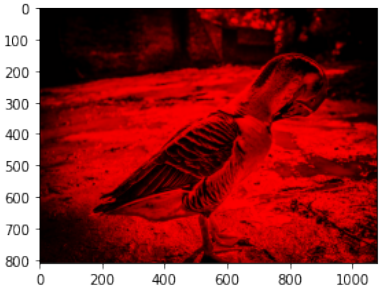

B channel
Height 810
Width 1080
0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000

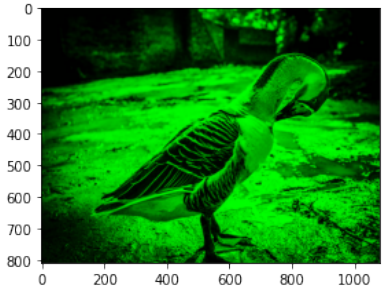

0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000

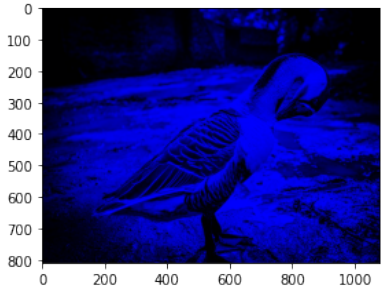

0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000

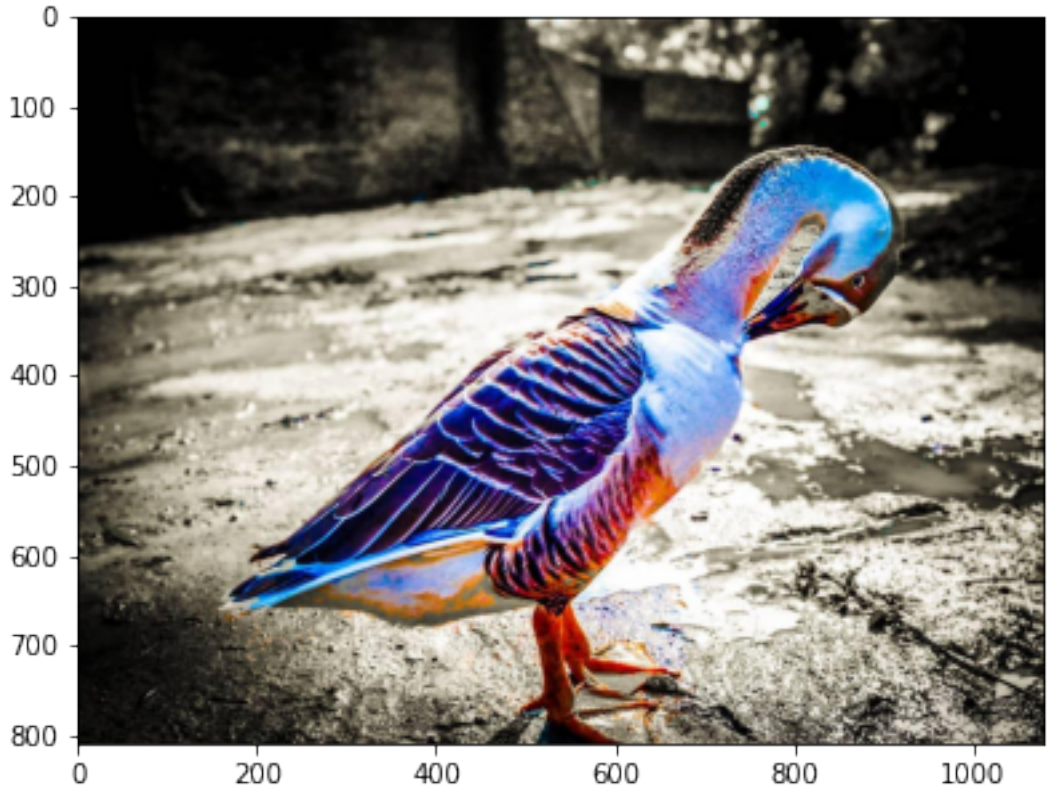

0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000


0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000


0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000


0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000


0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000


0
100
200
300
400
500
600
700
800
0
200
400
600
800
1000

11. Graphs (If Any): Image /Soft copy of graph paper to be attached here