OPENCV ASSIGNMENT REPORT

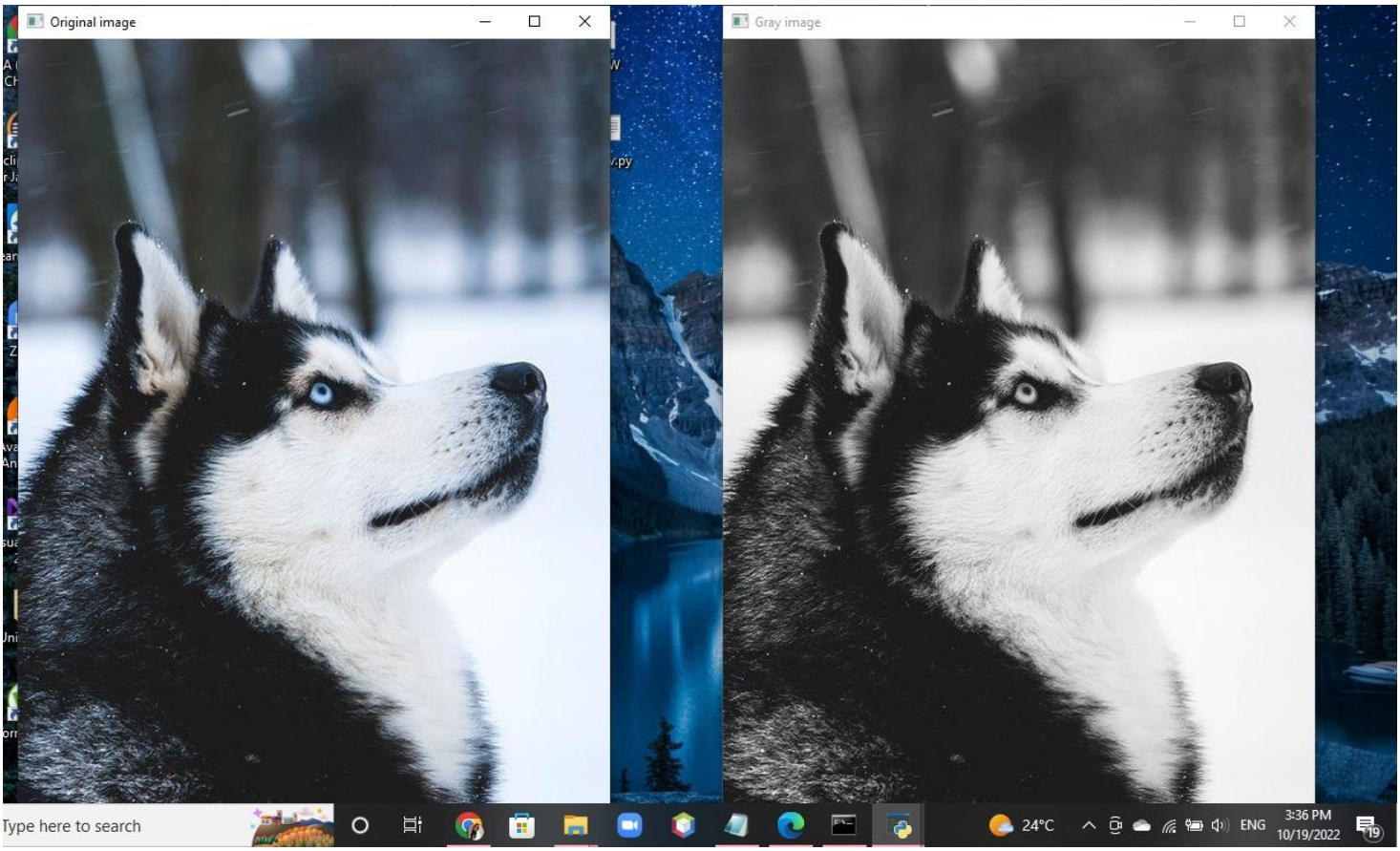
Made by : Jana khammash

---

1. Read a color image and Convert the image into gray-scale

CODE:

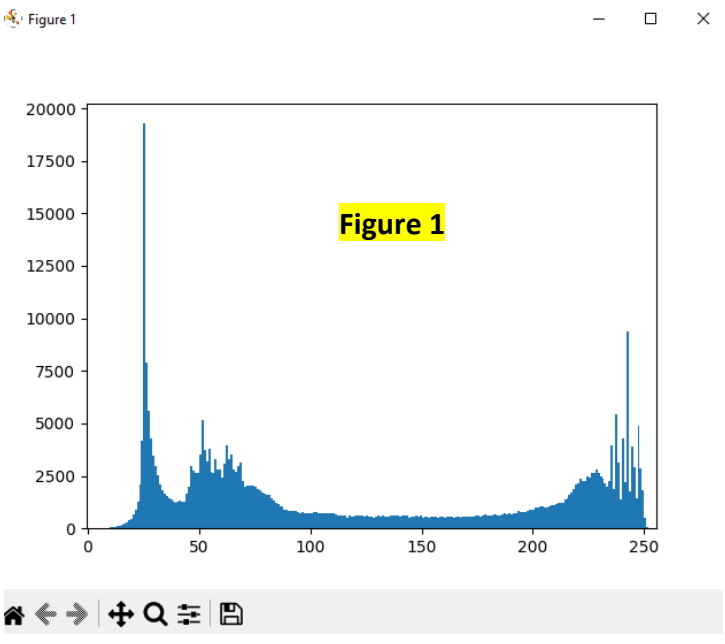
```
image = cv2.imread('C:/Users/hp/Desktop/dog.jpg')

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

cv2.imshow('Original image',image)

cv2.imshow('Gray image', gray)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

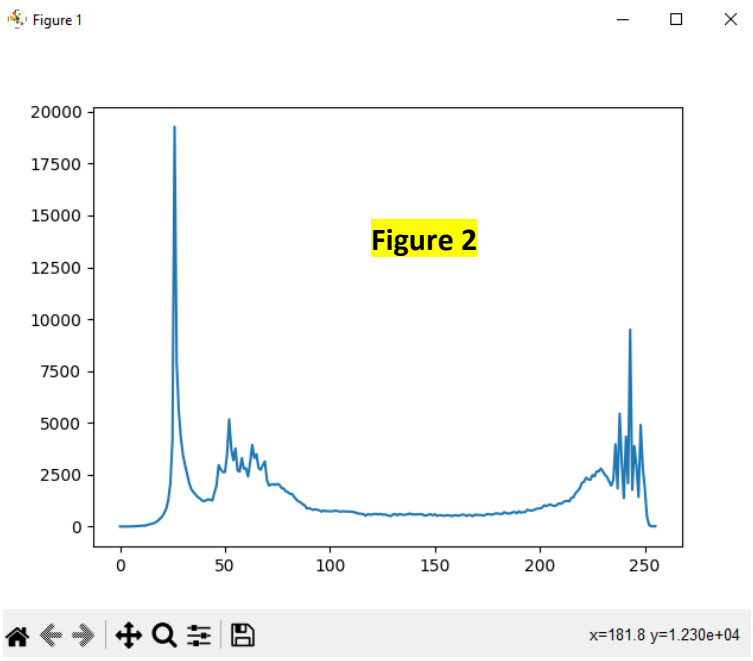


2. Show the histogram of the image

CODE :

```
im = cv2.imread('C:/Users/hp/Desktop/gray.jpg')
# calculate mean value from RGB channels and flatten to 1D array
 vals = im.mean(axis=2).flatten()
# calculate histogram
counts, bins = np.histogram(vals, range(257))
# plot histogram centered on values 0..255
plt.bar(bins[:-1] - 0.5, counts, width=1, edgecolor='none')
plt.xlim([-0.5, 255.5])
plt.show()   DRAW FIGURE 1
from matplotlib import pyplot as plt
img = cv2.imread('C:/Users/hp/Desktop/gray.jpg',0)
# find frequency of pixels in range 0-255
histr = cv2.calcHist([img],[0],None,[256],[0,256])
# show the plotting graph of an image
plt.plot(histr)
plt.show()   DRAW FIGURE 2
```





3. Print the necessary features/values that can be useful to identify the shape of the histogram

CODE:

```
def dir(skew):
    if skew > 0:
        return "right"
    elif skew < 0:
        return "left"
    else:
        return "normally"
print("Histogram features: ")
print("mean: " + str(np.mean(counts)) + ", mode = " + str(stats.mode(counts)[0][0]) + ", median = " + str(np.median(counts)) + ", skew = " + str(stats.skew(counts)) + ", direction = " + dir(stats.skew(counts)) + " skewed")
```

```
C:\Users\hp>python "C:/Users/hp/Desktop/HW1Jana.py"
Histogram features:
C:\Users\hp\Desktop\HW1Jana.py:32: FutureWarning: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, this behavior will change: the default value of `keepdims` will become False, the `axis` over which the statistic is taken will be eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avoid this warning.
  print("mean: " + str(np.mean(counts)) + ", mode = " + str(stats.mode(counts)[0][0]) + ", median = " + str(np.median(counts)) + ", skew = " + str(stats.skew(counts)) + ", direction = " + dir(stats.skew(counts)) + " skewed")
mean: 1433.90625, mode = 553, median = 788.5, skew = 5.4342749283946015, direction = right skewed

C:\Users\hp>
```

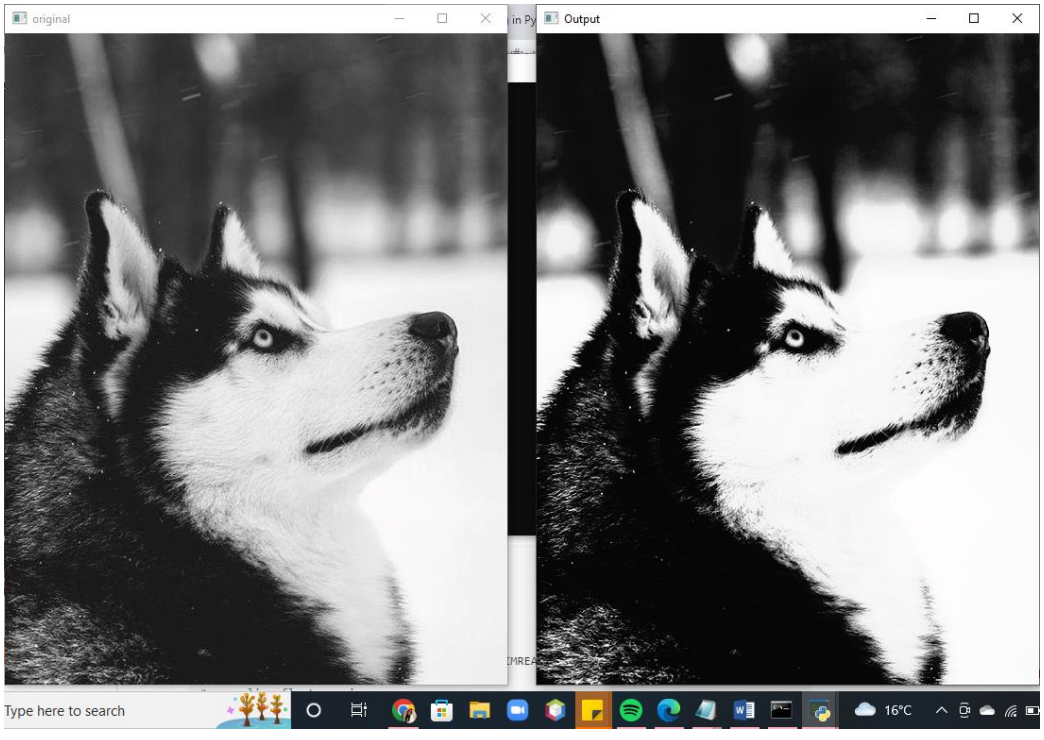
Output : mean: 1433.90625, mode = 553, median = 788.5, skew = 5.4342749283946015, direction = right skewed

4. Enhance the contrast of the image using the following techniques and compare between the resulting images

CODE:

- **Automatically :**

```
import cv2
import numpy as np
img = cv2.imread('C:/Users/hp/Desktop/gray.jpg')
original = img.copy()
xp = [0, 64, 128, 192, 255]
fp = [0, 16, 128, 240, 255]
x = np.arange(256)
table = np.interp(x, xp, fp).astype('uint8')
img = cv2.LUT(img, table)
cv2.imshow("original", original)
cv2.imshow("Output", img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



- **Manually :**

CODE:

```
from __future__ import print_function

from __future__ import division

import cv2 as cv

import numpy as np

import argparse

alpha = 1.0

alpha_max = 500

beta = 0

beta_max = 200

gamma = 1.0

gamma_max = 200

def basicLinearTransform():

res = cv.convertScaleAbs(img_original, alpha=alpha, beta=beta)

img_corrected = cv.hconcat([img_original, res])

cv.imshow("Brightness and contrast adjustments", img_corrected)

def gammaCorrection():

## [changing-contrast-brightness-gamma-correction]

lookUpTable = np.empty((1,256), np.uint8)

for i in range(256):

lookUpTable[0,i] = np.clip(pow(i / 255.0, gamma) * 255.0, 0, 255)

res = cv.LUT(img_original, lookUpTable)

## [changing-contrast-brightness-gamma-correction]
```

```
img_gamma_corrected = cv.hconcat([img_original, res])

cv.imshow("Gamma correction", img_gamma_corrected)

def on_linear_transform_alpha_trackbar(val):

global alpha

alpha = val / 100

basicLinearTransform()

def on_linear_transform_beta_trackbar(val):

global beta

beta = val - 100

basicLinearTransform()

def on_gamma_correction_trackbar(val):

global gamma

gamma = val / 100

gammaCorrection()

parser = argparse.ArgumentParser(description='Code for Changing the contrast and brightness of an image! tutorial.')

parser.add_argument('--input', help='Path to input image.', default='C:/Users/hp/Desktop/gray.jpg')

args = parser.parse_args()

img_original = cv.imread(cv.samples.findFile(args.input))

if img_original is None:

print('Could not open or find the image: ', args.input)

exit(0)

img_corrected = np.empty((img_original.shape[0], img_original.shape[1]*2, img_original.shape[2]), img_original.dtype)

img_gamma_corrected = np.empty((img_original.shape[0], img_original.shape[1]*2, img_original.shape[2]), img_original.dtype)
```

img_corrected = cv.hconcat([img_original, img_original])

img_gamma_corrected = cv.hconcat([img_original, img_original])

cv.namedWindow('Brightness and contrast adjustments')

cv.namedWindow('Gamma correction')

alpha_init = int(alpha *100)

cv.createTrackbar('Alpha gain (contrast)', 'Brightness and contrast adjustments', alpha_init, alpha_max, on_linear_transform_alpha_trackbar)

beta_init = beta + 100

cv.createTrackbar('Beta bias (brightness)', 'Brightness and contrast adjustments', beta_init, beta_max, on_linear_transform_beta_trackbar)

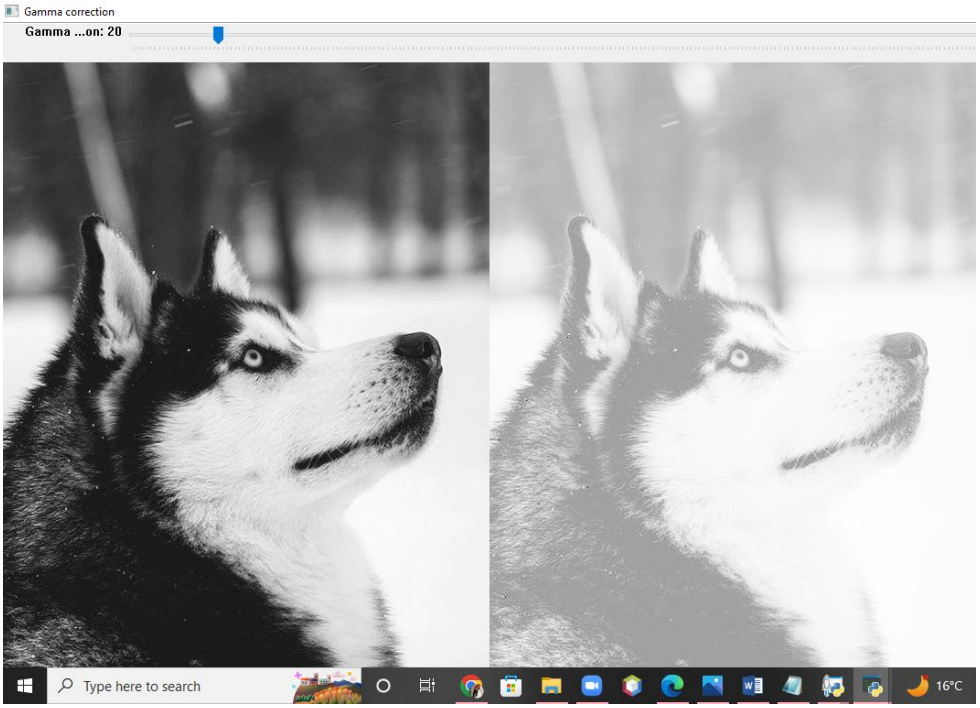
gamma_init = int(gamma * 100)

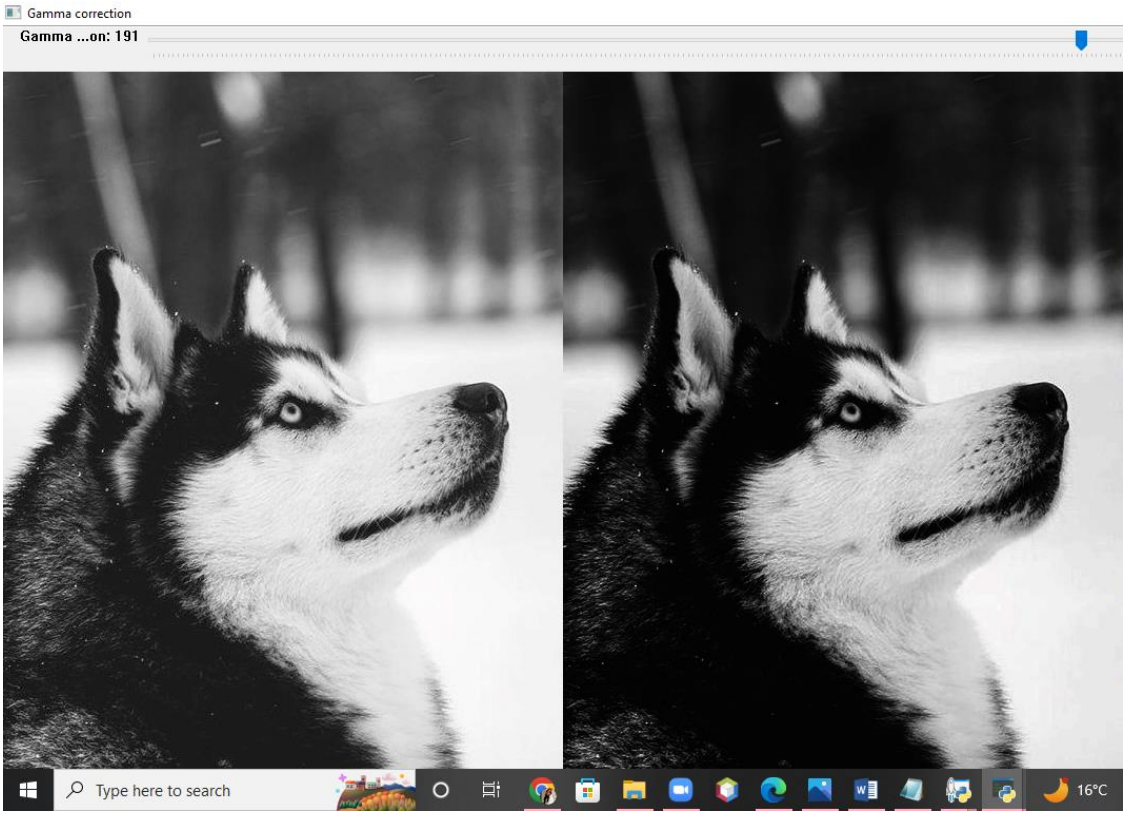
cv.createTrackbar('Gamma correction', 'Gamma correction', gamma_init, gamma_max, on_gamma_correction_trackbar)

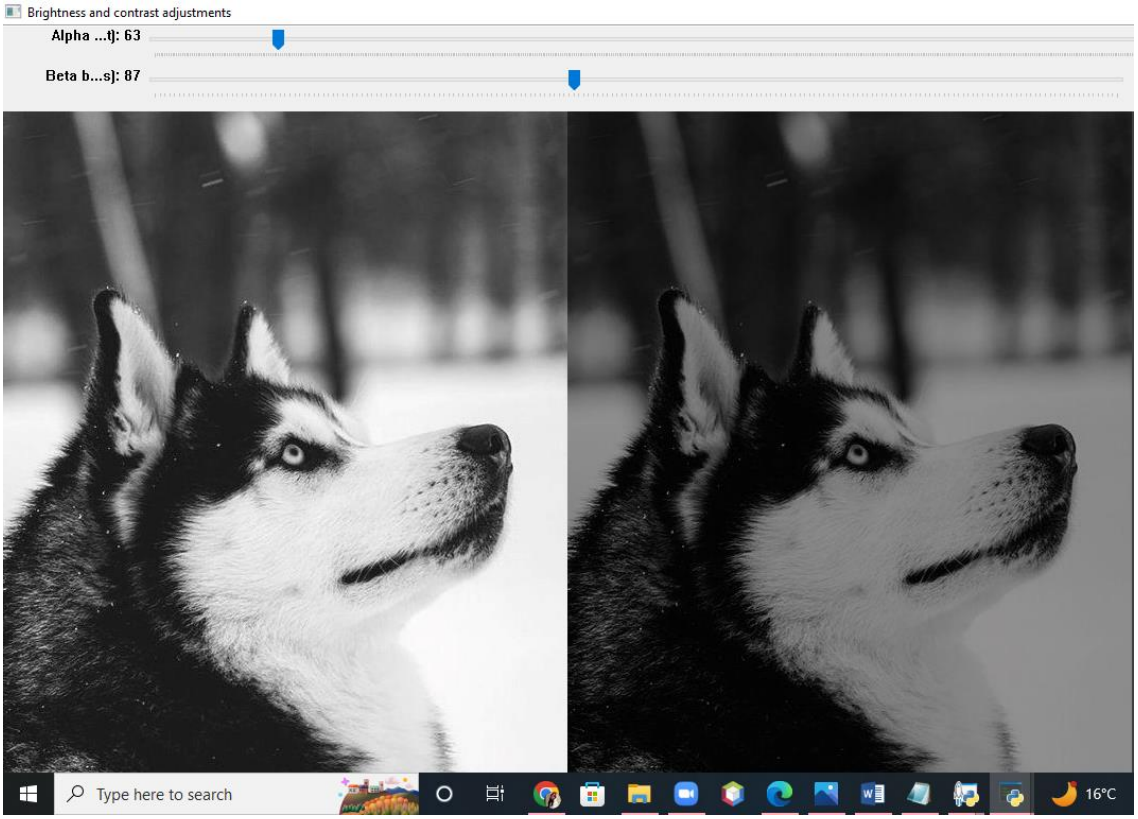
on_linear_transform_alpha_trackbar(alpha_init)

on_gamma_correction_trackbar(gamma_init)

cv.waitKey()




Gamma correction
Gamma ...on: 191
Type here to search
16°C


Brightness and contrast adjustments
Alpha ...t]: 63
Beta b...s]: 87
Type here to search
16°C