**CV Practical No.: 2**

**Aim: Understanding geometric translation and histogram**

**New Concept:**

**i. np.clip:** it is a NumPy function that limits the values in an array to a specified range. Any values lower than the minimum are set to the minimum, and values higher than the maximum are set to the maximum.

**ii. image.astype(float):** it is used to convert an image or array to a specific data type, in this case, a float. This is useful when you need more precision in calculations, such as for image transformations, processing, or normalization.

**iii. np.uint8:** it is a NumPy data type that stands for **unsigned 8-bit integer.** It can store integer values from 0 to 255. This is the most common data type for images, where pixel values typically range from 0 to 255.

**iv. warpAffine:** it is a function in OpenCV that applies an affine transformation to an image. An affine transformation preserves points, straight lines, and planes, but it can perform operations like rotation, scaling, translation, and shearing.

**v. calcHist**: it is a function in OpenCV that calculates the histogram of an image or a region of an image. A histogram is a graphical representation of the distribution of pixel intensities in an image.

**vi. equalizeHist:** it is a function in OpenCV that performs histogram equalization on an image. This technique enhances the contrast of an image by spreading out the most frequent intensity values. It is used to improve the quality of images with poor contrast.

**Theory:**

**Translation:**

**Translation** refers to the process of shifting an image or object in space along the horizontal (x-axis) and/or vertical (y-axis) direction without changing its content. The object or image is moved by a certain number of pixels or units in a given direction.

**Histogram:**

A histogram in computer vision is a graphical representation of the distribution of pixel intensities in an image. It shows how many pixels in the image have each possible intensity value (from 0 to 255 for grayscale images).

* **For grayscale images:** The x-axis of the histogram represents pixel intensity values (ranging from 0 for black to 255 for white), while the y-axis represents the frequency or count of pixels with those intensities.
* **For color images:** The histogram is typically calculated separately for each color channel (Red, Green, and Blue).

**Program:**

**Program1: to convert original image to brightened, dark, rotate and translate image:**

import cv2

from scipy import ndimage

import matplotlib.pyplot as plt

import numpy as np

#Load the image

image = cv2.imread("C:/Users/admin/Desktop/Nupur/flower.jpg")

#Increase Brightness

image\_float = image.astype(float)

# Increase the brightness (values greater than 1 between 2 to 5)

brightness\_factor = 2.5 # Increase or decrease this value as needed

brightened\_image = image\_float \* brightness\_factor

#Decrease the Brightness (values between 0 and 1)

brightness\_factor1 = 0.4 # Increase or decrease this value as needed

dark\_image = image\_float \* brightness\_factor1

# Clip the pixel values to the valid range [0, 255]

brightened\_image = np.clip(brightened\_image, 0, 255)

dark\_image = np.clip(dark\_image, 0, 255)

# Convert the image back to unsigned 8-bit integers

brightened\_image = brightened\_image.astype(np.uint8)

dark\_image = dark\_image.astype(np.uint8)

# Image Rotation

image\_rotate = ndimage.rotate(image,45)

#Image translation

height, width = image.shape[:2] #Store height and Weight

T = np.float32([[1, 0, 100], [0, 1, 200]])

img\_trans = cv2.warpAffine(image, T, (width, height))

#Show images

#cv2.imshow("Original Image",image)

#cv2.imshow("Brightened image",brightened\_image)

#cv2.imshow("Dark image",dark\_image)

#cv2.imshow("Rotated image",image\_rotate)

#cv2.imshow("Translated image",img\_trans)

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

#Image1

plt.subplot(2,3,1)

plt.imshow(image)

plt.title('Original Image')

plt.axis('off')

#Image2

plt.subplot(2,3,2)

plt.imshow(brightened\_image)

plt.title('Brightened Image')

plt.axis('off')

#Image3

plt.subplot(2,3,3)

plt.imshow(dark\_image)

plt.title('Dark Image')

plt.axis('off')

#Image4

plt.subplot(2,3,4)

plt.imshow(image\_rotate)

plt.title('Rotated Image')

plt.axis('off')

#Image5

plt.subplot(2,3,5)

plt.imshow(img\_trans)

plt.title('Translated Image')

plt.axis('off')

plt.show()

#cv2.waitKey(0)

#cv2.destroyAllWindows()

**Program2: histogram plotting:**

import cv2

import matplotlib.pyplot as plt

# Step 2: Read the image

image = cv2.imread("C:/Users/admin/Desktop/Nupur/sunset.jpg")

# Step 3: Convert to grayscale (optional)

gray\_img = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# Step 4: Calculate the histogram

hist = cv2.calcHist([gray\_img], [0], None, [256], [0, 256])

# Step 5: Plot the histogram (optional)

plt.plot(hist)

plt.title('Histogram of Grayscale Image')

plt.xlabel('Pixel Intensity')

plt.ylabel('Frequency')

plt.show()

**Program3: histogram equalization:**

#import required Libraries

import cv2

from scipy import ndimage

import numpy as np

import matplotlib.pyplot as plt

#Load the image

image = cv2.imread("C:/Users/admin/Desktop/Nupur/sunset.jpg")

#Convert to Grayscale

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

#Perform Histogram Equalization

equalized\_img = cv2.equalizeHist(gray)

#cv2.imshow("Original Gray Image",gray)

#cv2.imshow("Histogram Equalized Image",equalized\_img)

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

#Image1

plt.subplot(2,2,1)

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

plt.title('Original Gray Image')

plt.axis('off')

#Image2

plt.subplot(2,2,2)

plt.imshow(equalized\_img, cmap = 'gray')

plt.title('Histogram Equalized Image')

plt.axis('off')

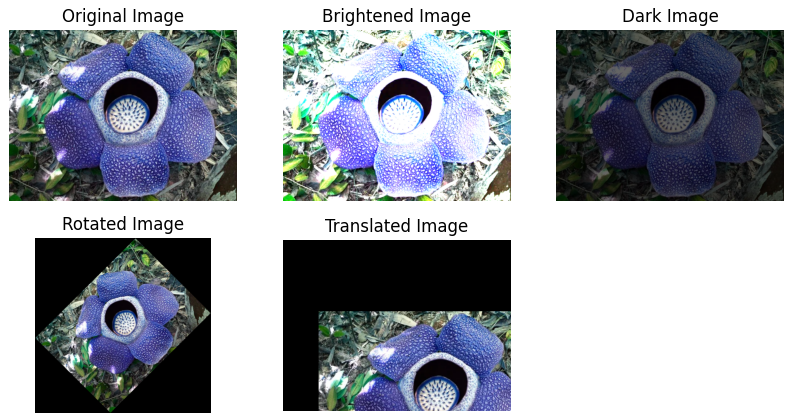
plt.show()

#cv2.waitKey(0)

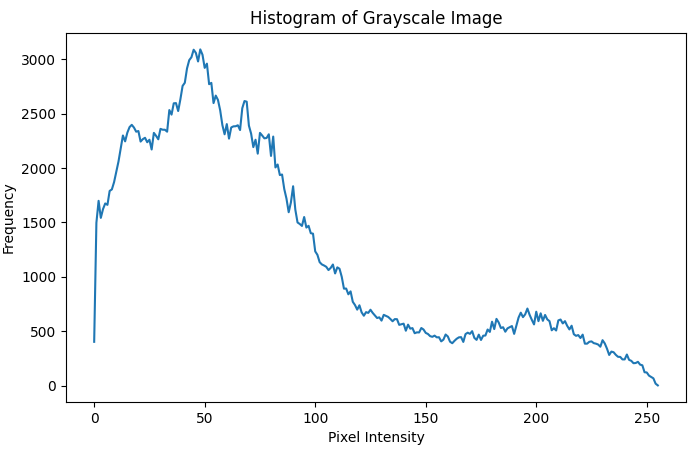
#cv2.destroyAllWindows()

**Output:**

**Program 1:**



**Program 2:**



**Program 3:**

