* **Lab-1** **Image Enhancement Techniques using OpenCV – Python—**

**IMAGE ENHANCEMENT**

It is the process of image manipulation to make it more suitable for specific use. It provides better contrast and a more detailed image and is used to enhance medical images, images captured in remote sensing, images from satellite etc.

Image enhancement is the process of improving the quality and appearance of an image. It can be used to correct flaws or defects in an image, or to simply make an image more visually appealing. Image enhancement techniques can be applied to a wide range of images, including photographs, scans, and digital images. Some common goals of image enhancement include increasing contrast, sharpness, and colorfulness; reducing noise and blur; and correcting distortion and other defects. Image enhancement techniques can be applied manually using image editing software, or automatically using algorithms and computer programs such as OpenCV.   
In this article, we will explore a variety of image enhancement techniques that can be performed using OpenCV and Python. OpenCV is a powerful, open-source computer vision library that provides a wide range of image processing and computer vision algorithms. By combining the capabilities of OpenCV with the versatility of Python, we can easily implement a variety of image enhancement techniques to improve the quality and appearance of our images.

**Getting started**

Prerequisites for running the code are:

1. Python
2. Python-OpenCV Library
3. Jupiter Notebook

**Image Enhancement can be done in:**

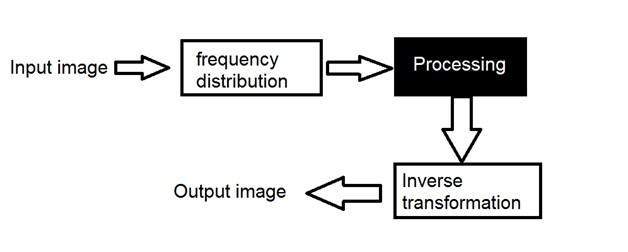
* **Spatial Domain:**

There is a direct manipulation of a pixel in an image (on the image plane)



* **Frequency Domain:**

Processing the image is based on modifying the Fourier transform of an image.



**Why is it done?**

It is done for the following reasons:

* Highlighting interesting detail in images.
* Removing noise from images.
* Making images more visually appealing.

The transformation function is given as follows:

**S = T(r)**

Where, r is the pixels of the input image. s is the pixels of the output image. T is a transformation function that maps each value to r to each value of s.

It can be achieved through grey level transformations.

1. **Adjusting brightness and contrast**

There are several ways to adjust the brightness and contrast of an image using OpenCV and Python. One common method is to use the **cv2.addWeighted()** function, which allows you to adjust the brightness by adding a scalar value to each pixel in the image, and the contrast by scaling the pixel values.

#Import the necessary libraries

import cv2

import matplotlib.pyplot as plt

import numpy as np

# Load the image

image = cv2.imread('GFG.jpeg')

#Plot the original image

plt.subplot(1, 2, 1)

plt.title("Original")

plt.imshow(image)

# Adjust the brightness and contrast

# Adjusts the brightness by adding 10 to each pixel value

brightness = 10

# Adjusts the contrast by scaling the pixel values by 2.3

contrast = 2.3

image2 = cv2.addWeighted(image, contrast, np.zeros(image.shape, image.dtype), 0, brightness)

#Save the image

cv2.imwrite('modified\_image.jpg', image2)

#Plot the contrast image

plt.subplot(1, 2, 2)

plt.title("Brightness & contrast")

plt.imshow(image2)

plt.show()

1. **Sharpening images**

Sharpening is the process of enhancing the edges and fine details in an image to make it appear sharper and more defined. It is important because it can help to bring out the details and features in an image, making it more visually appealing and easier to understand.

One common method for sharpening images using OpenCV and Python is to use the **cv2.filter2D()** function, which convolves the image with a kernel. The kernel can be designed to enhance the edges in the image, resulting in a sharper image.

#Import the necessary libraries

import cv2

import matplotlib.pyplot as plt

import numpy as np

# Load the image

image = cv2.imread('GFG.jpeg')

#Plot the original image

plt.subplot(1, 2, 1)

plt.title("Original")

plt.imshow(image)

# Create the sharpening kernel

kernel = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])

# Sharpen the image

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

#Save the image

cv2.imwrite('sharpened\_image.jpg', sharpened\_image)

#Plot the sharpened image

plt.subplot(1, 2, 2)

plt.title("Sharpening")

plt.imshow(sharpened\_image)

plt.show()

1. **Enhancing color in images**

Color enhancement is adjusting the colors in an image to make them more vibrant, balanced, or natural.

There are several ways to enhance the colors in an image using OpenCV and Python. One common method is to use the **cv2.cvtColor()** function, which allows you to convert the image from one color space to another. This can be useful for adjusting the color balance or saturation of the image.  
Here is an example of how to enhance the colors in an image using the **cv2.cvtColor()** function:

#Import the necessary libraries

import cv2

import matplotlib.pyplot as plt

import numpy as np

# Load the image

image = cv2.imread('GFG.jpeg')

#Plot the original image

plt.subplot(1, 2, 1)

plt.title("Original")

plt.imshow(image)

# Convert the image from BGR to HSV color space

image = cv2.cvtColor(image, cv2.COLOR\_RGB2HSV)

# Adjust the hue, saturation, and value of the image

# Adjusts the hue by multiplying it by 0.7

image[:, :, 0] = image[:, :, 0] \* 0.7

# Adjusts the saturation by multiplying it by 1.5

image[:, :, 1] = image[:, :, 1] \* 1.5

# Adjusts the value by multiplying it by 0.5

image[:, :, 2] = image[:, :, 2] \* 0.5

# Convert the image back to BGR color space

image2 = cv2.cvtColor(image, cv2.COLOR\_HSV2BGR)

#Save the image

cv2.imwrite('enhanced coloured.jpg', image2)

#Plot the enhanced image

plt.subplot(1, 2, 2)

plt.title("enhanced coloured")

plt.imshow(image2)

plt.show()

**References-**

<https://www.geeksforgeeks.org/image-enhancement-techniques-using-opencv-python/>

<https://ninjakx.github.io/Image_Enhancement/>

<https://www.kaggle.com/code/dilipkumar2k6/opencv-module-4-image-enhancement>