DIP Assignment - I

Image Enhancement

By : Prafful Mishra [RA2011003010802]
Subham Gayen [RA2011003010798]

Section : CSE - B2

Submitted To : Dr. K. Alice

1. Image Enhancement Techniques

Introduction

Image upscaling is the process of enlarging image without any loss in its quality. This makes the image presentable in larger formats. It is an essential part of image processing as, images need to be scaled up or down for multiple reasons.

The principal objective of image enhancement is to process a given image so that the result is more suitable than the original image for a specific application. It accentuates or sharpens image features such as edges, boundaries, or contrast to make a graphic display more helpful for display and analysis. The enhancement doesn't increase the inherent information content of the data, but it increases the dynamic range of the chosen features so that they can be detected easily. The greatest difficulty in image enhancement is quantifying the criterion for enhancement and, therefore, a large number of image enhancement techniques are empirical and require interactive procedures to obtain satisfactory results. Image enhancement methods can be based on either spatial or frequency domain techniques.

Image Enhancement Techniques

a) Sharpening Images :

It 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.

Sharpening can be used to correct blur or softness in an image and can be applied using a variety of techniques. 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.

Source Code

```
# Import the necessary libraries
import cv2
import matplotlib.pyplot as plt
import numpy as np

# Load the image
Lenna = cv2.imread('test_img1.png')

# Plot the original image
plt.subplot(1, 2, 1)
plt.title("Original")
plt.imshow(Lenna)
```

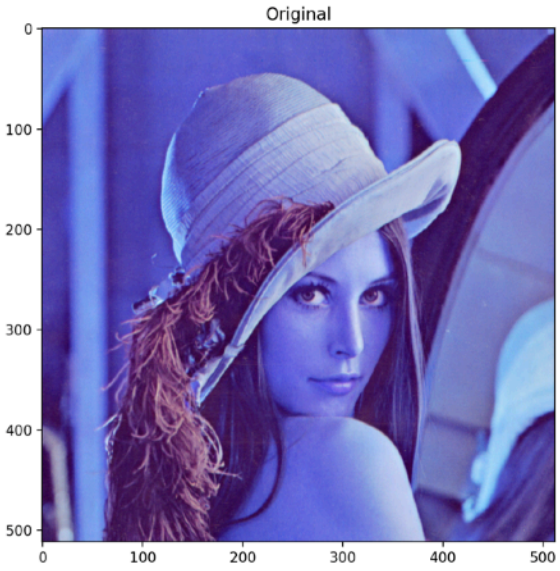
```
# Create the sharpening kernel
x = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])

# Sharpen the image
Lemma2 = cv2.filter2D(Lenna, -1, x)

# Save the image
cv2.imwrite('SharpenImage.jpg', Lemma2)

# Plot the sharpen image
plt.subplot(1, 2, 2)
plt.title("Sharpen")
plt.imshow(Lemma2)
plt.show()
```

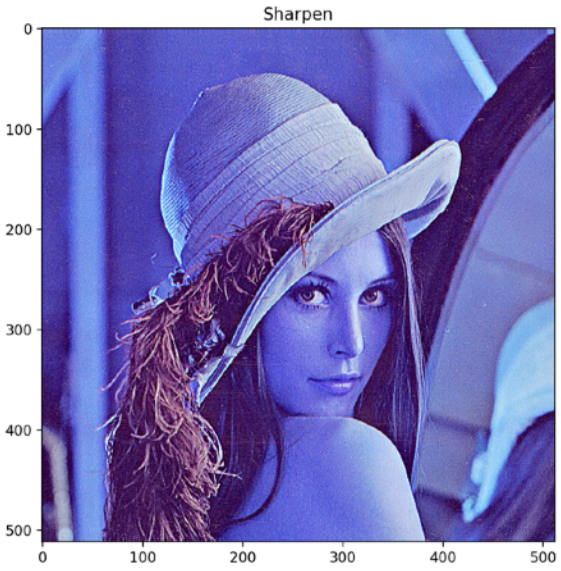
Program

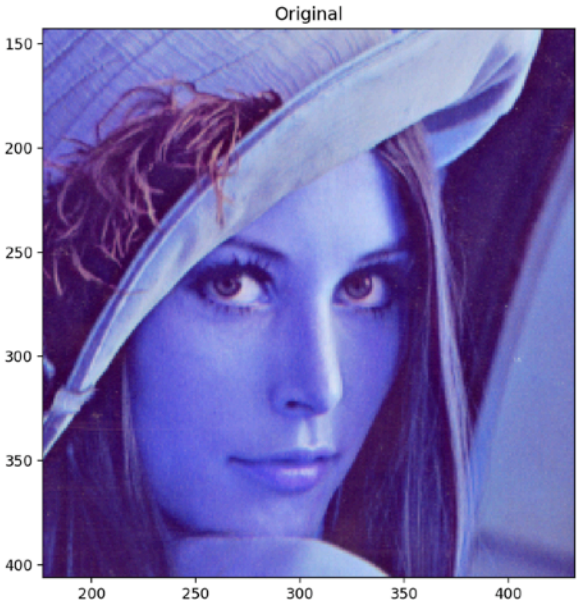
ImageSharpening.py

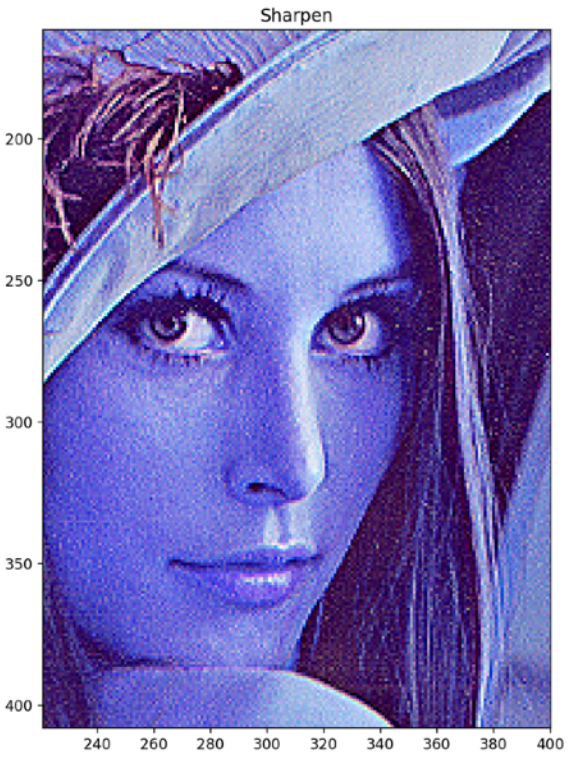
```python
# Import the necessary libraries
import cv2
import matplotlib.pyplot as plt
import numpy as np

# Load the image
Lenna = cv2.imread('test_img1.png')

# Plot the original image
plt.subplot(1, 2, 1)
plt.title("Original")
plt.imshow(Lenna)


# Create the sharpening kernel
x = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])

# Sharpen the image
Lemma2 = cv2.filter2D(Lenna, -1, x)

# Save the image
cv2.imwrite('SharpenImage.jpg', Lemma2)

# Plot the sharpen image
plt.subplot(1, 2, 2)
plt.title("Sharpen")
plt.imshow(Lemma2)
plt.show()

```

Output


Original
0
100
200
300
400
500
0
100
200
300
400
500


Sharpen
0
100
200
300
400
500
0
100
200
300
400
500


Original
150
200
250
300
350
400
200
250
300
350
400


Sharpen
200
250
300
350
400
240
260
280
300
320
340
360
380
400

b) Enhancing Color in Images :

Colour enhancement is adjusting the colours in an image to make them more vibrant, balanced, or natural. It can be used to correct colour defects or problems in an image or to simply make an image more appealing and aesthetically pleasing. Colour enhancement is important because it can significantly affect the visual impact and effectiveness of an image. It can also be useful for correcting colour errors or problems in an image and can make it easier to see details and features in the image. There are several techniques for enhancing the colours in an image, including adjusting the colour balance, adjusting the saturation, and adjusting the hue.

There are several ways to enhance the colours 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 colour space to another. This can be useful for adjusting the colour balance or saturation of the image.

Source Code

```
# Import the necessary libraries
import cv2
import matplotlib.pyplot as plt
import numpy as np

# Load the image
Lenna = cv2.imread('test_img1.png')

# Plot the original image
plt.subplot(1, 2, 1)
plt.title("Original")
plt.imshow(Lenna)

# Convert the image from BGR to HSV color space
Lenna = cv2.cvtColor(Lenna, cv2.COLOR_RGB2HSV)
```

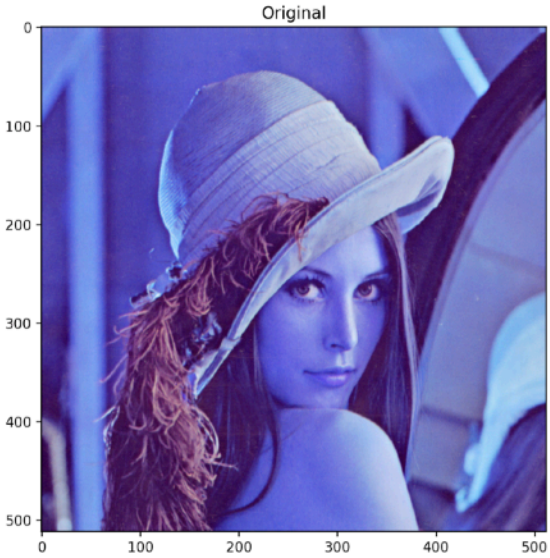
```
# Adjust the hue, saturation, and value of the image
# Adjusts the hue by multiplying it by 0.7
Lenna[:, :, 0] = Lenna[:, :, 0] * 0.7
# Adjusts the saturation by multiplying it by 1.5
Lenna[:, :, 1] = Lenna[:, :, 1] * 1.5
# Adjusts the value by multiplying it by 0.5
Lenna[:, :, 2] = Lenna[:, :, 2] * 0.5

# Convert the image back to BGR color space
Lenna2 = cv2.cvtColor(Lenna, cv2.COLOR_HSV2BGR)

# Save the image
cv2.imwrite('EnhancedColoured.jpg', Lenna2)

# Plot the enhanced image
plt.subplot(1, 2, 2)
plt.title("Enhanced Coloured")
plt.imshow(Lenna2)
plt.show()
```

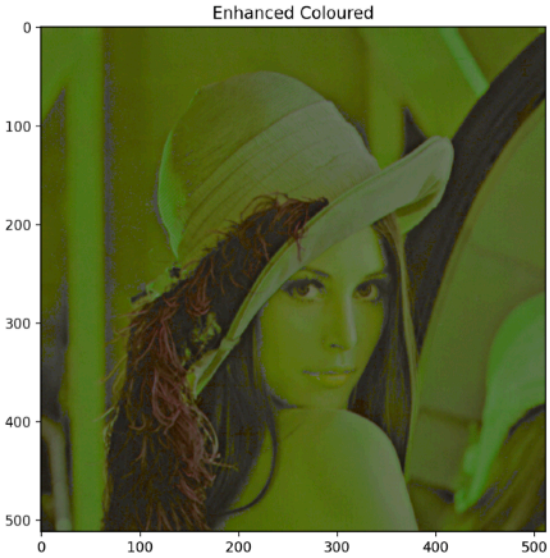
Program :

ColourEnhancement.py

```python
# Import the necessary libraries
import cv2
import matplotlib.pyplot as plt
import numpy as np

# Load the image
Lenna = cv2.imread('test_img1.png')

# Plot the original image
plt.subplot(1, 2, 1)
plt.title("Original")
plt.imshow(Lenna)

# Convert the image from BGR to HSV color space
Lenna = cv2.cvtColor(Lenna, cv2.COLOR_RGB2HSV)

# Adjust the hue, saturation, and value of the image
# Adjusts the hue by multiplying it by 0.7
Lenna[:, :, 0] = Lenna[:, :, 0] * 0.7
# Adjusts the saturation by multiplying it by 1.5
Lenna[:, :, 1] = Lenna[:, :, 1] * 1.5
# Adjusts the value by multiplying it by 0.5
Lenna[:, :, 2] = Lenna[:, :, 2] * 0.5

# Convert the image back to BGR color space
Lenna2 = cv2.cvtColor(Lenna, cv2.COLOR_HSV2BGR)

# Save the image
cv2.imwrite('EnhancedColoured.jpg', Lenna2)

# Plot the enhanced image
plt.subplot(1, 2, 2)
plt.title("Enhanced Coloured")
plt.imshow(Lenna2)
plt.show()

```

Output


Original
0
100
200
300
400
500
0
100
200
300
400
500


Enhanced Coloured
0
100
200
300
400
500
0
100
200
300
400
500


Original
200
225
250
275
300
325
350
375
400
225
250
275
300
325
350
375


Enhanced Coloured
200
225
250
275
300
325
350
375
400
220
240
260
280
300
320
340
360

2. Histogram Equalisation

Introduction

Histogram Equalisation is a computer image processing technique used to improve contrast in images. It accomplishes this by effectively spreading out the most frequent intensity values, i.e. stretching out the intensity range of the image. This method usually increases the global contrast of images when its usable data is represented by close contrast values. This allows for areas of lower local contrast to gain a higher contrast.

A colour histogram of an image represents the number of pixels in each type of colour component. Histogram equalisation cannot be applied separately to the Red, Green and Blue components of the image as it leads to dramatic changes in the image's colour balance. However, if the image is first converted to another colour space, like HSL/HSV colour space, then the algorithm can be applied to the luminance or value channel without resulting in changes to the hue and saturation of the image.

Source Code

```
# import Libraries
import cv2
import numpy as np

# read a image using imread
Lenna = cv2.imread('test_img2.png', 0)

# creating a Histograms Equalization
# of a image using cv2.equalizeHist()
a = cv2.equalizeHist(Lenna)

# stacking images side-by-side
x = np.hstack((Lenna, a))

# show image input vs output
cv2.imshow('Equalised', x)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Program

HistogramEquilisation.py

```python
# import Libraries
import cv2
import numpy as np

# read a image using imread
Lenna = cv2.imread('test_img2.png', 0)

# creating a Histograms Equalization
# # of a image using cv2.equalizeHist()
a = cv2.equalizeHist(Lenna)

# stacking images side-by-side
x = np.hstack((Lenna, a))

# show image input vs output
cv2.imshow('Equalised', x)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output

Results

We used image sharpening and colour enhancement techniques to improve the quality of the image by highlighting the required features and extracting them for further processing.

In histogram equalisation, we took a grayscale image as our input and adjusted the contrast by using the image's histogram.

The input, i.e., 'test_img1.png' and 'test_img2.png' (coloured and a grayscale image, respectively), was a 512x512-pixel image of "Lenna", which is a standard test image used in the field of digital image processing.

The output showed the comparison between the original and enhanced image by plotting each image next to the other, and the results were significantly good.

Hence, the image enhancement was successfully executed on the input image using Python code.