MACHINE VISION LAB

Lab Ex 2: Gray Level Transformation – Image Negative, Gamma Correction, Log transform, Image Enhancement - Low pass/High Pass spatial Filtering – Gaussian Filters, Noise Filtering.

NAME : OM SUBRATO DEY

REG NO.: 21BAI1876

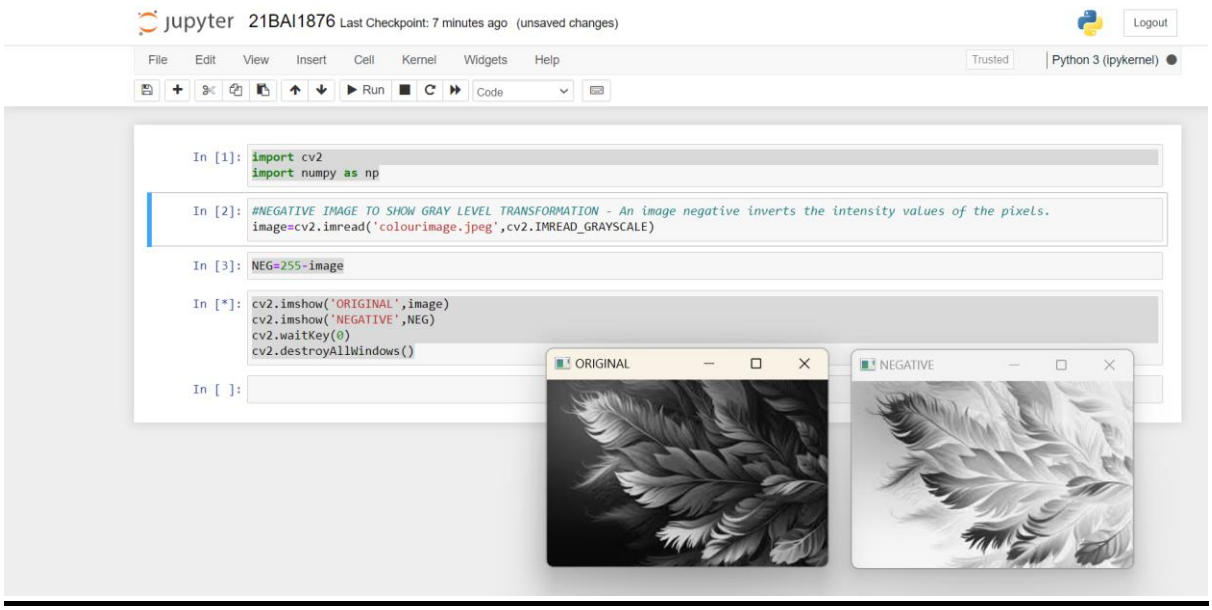
1. Gray Level Transformation – Image Negative

[An image negative inverts the intensity values of the pixels]

CODE:

```
import cv2

import numpy as np


#NEGATIVE IMAGE TO SHOW GRAY LEVEL TRANSFORMATION

image=cv2.imread('colourimage.jpeg',cv2.IMREAD_GRAYSCALE)

NEG=255-image

cv2.imshow('ORIGINAL',image)

cv2.imshow('NEGATIVE',NEG)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

OUTPUT:



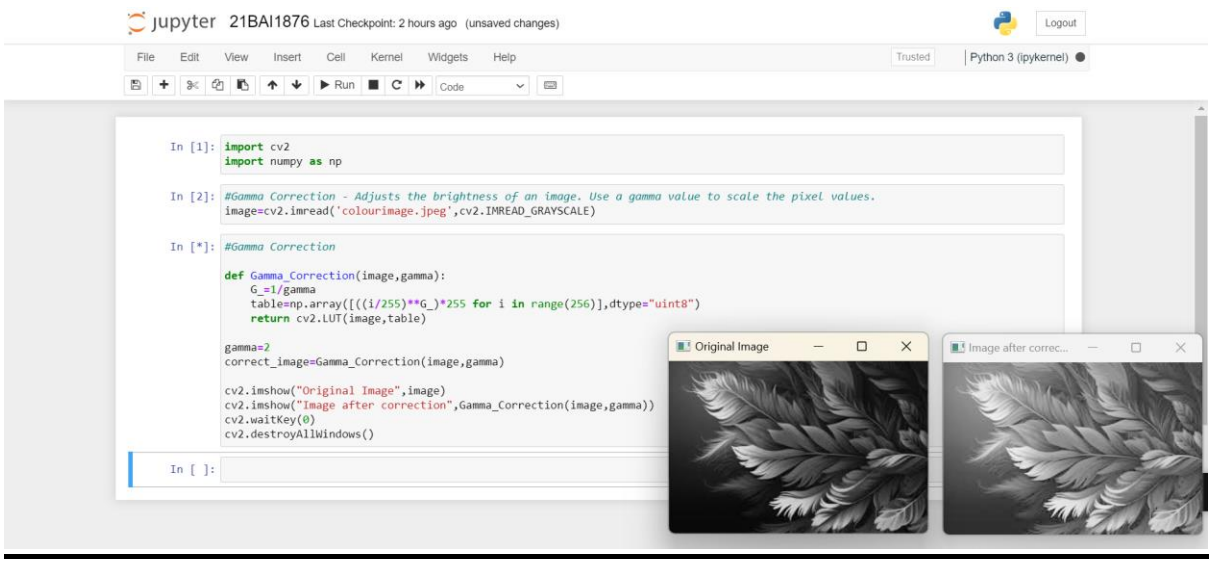
2. Gamma Correction – Modifying Brightness

[Adjusts the brightness of an image. Use a gamma value to scale the pixel values.]

CODE:

```
import cv2
import numpy as np
#reading image
image=cv2.imread('colourimage.jpeg',cv2.IMREAD_GRAYSCALE)


def Gamma_Correction(image,gamma):
    G_=1/gamma
    table=np.array([((i/255)**G_)*255 for i in range(256)],dtype="uint8")
    return cv2.LUT(image,table)


gamma=2
correct_image=Gamma_Correction(image,gamma)


cv2.imshow("Original Image",image)
cv2.imshow("Image after correction",Gamma_Correction(image,gamma))
cv2.waitKey(0)
cv2.destroyAllWindows()
```

OUTPUT:


jupyter 21BAI1876 Last Checkpoint: 2 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [1]:
import cv2
import numpy as np
In [2]:
#Gamma Correction - Adjusts the brightness of an image. Use a gamma value to scale the pixel values.
image=cv2.imread('colourimage.jpeg',cv2.IMREAD_GRAYSCALE)
In [*]:
#Gamma Correction
def Gamma_Correction(image,gamma):
G_=1/gamma
table=np.array([((i/255)**G_)*255 for i in range(256)],dtype="uint8")
return cv2.LUT(image,table)
gamma=2
correct_image=Gamma_Correction(image,gamma)
cv2.imshow("Original Image",image)
cv2.imshow("Image after correction",Gamma_Correction(image,gamma))
cv2.waitKey(0)
cv2.destroyAllWindows()
In [ ]:
Original Image
Image after correc...

3. Log transform – For better dynamic range

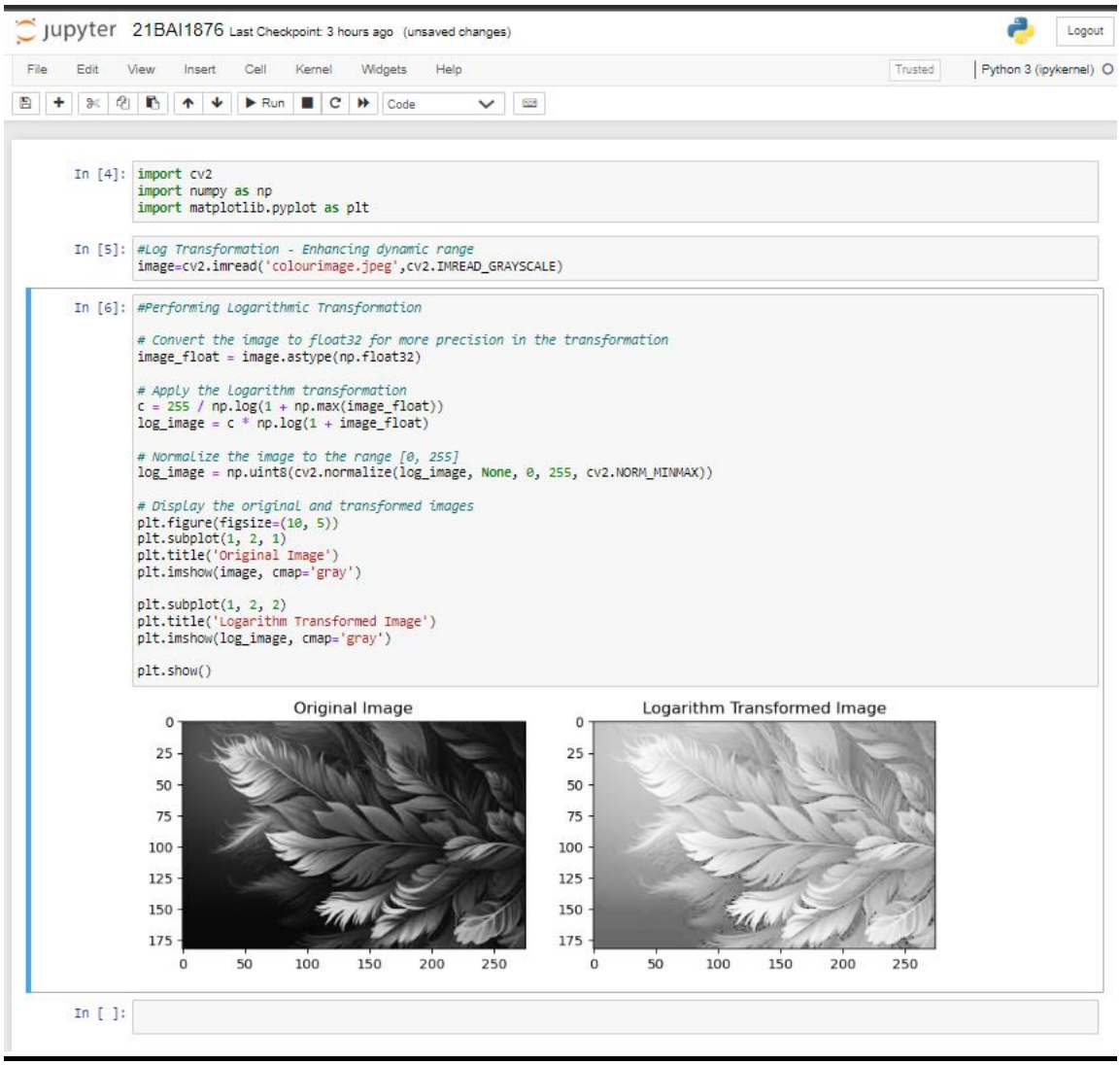
[Enhances the dynamic range of an image.]

CODE:

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

#Log Transformation - Enhancing dynamic range
image=cv2.imread('colourimage.jpeg',cv2.IMREAD_GRAYSCALE)

#Performing Logarithmic Transformation
# Convert the image to float32 for more precision in the transformation
image_float = image.astype(np.float32)

# Apply the logarithm transformation
c = 255 / np.log(1 + np.max(image_float))
log_image = c * np.log(1 + image_float)

# Normalize the image to the range [0, 255]
log_image = np.uint8(cv2.normalize(log_image, None, 0, 255, cv2.NORM_MINMAX))

# Display the original and transformed images
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.title('Original Image')
```

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


plt.subplot(1, 2, 2)
plt.title('Logarithm Transformed Image')
plt.imshow(log_image, cmap='gray')
plt.show()
```

OUTPUT:



4. Image Enhancement - Low Pass/High Pass Spatial Filtering

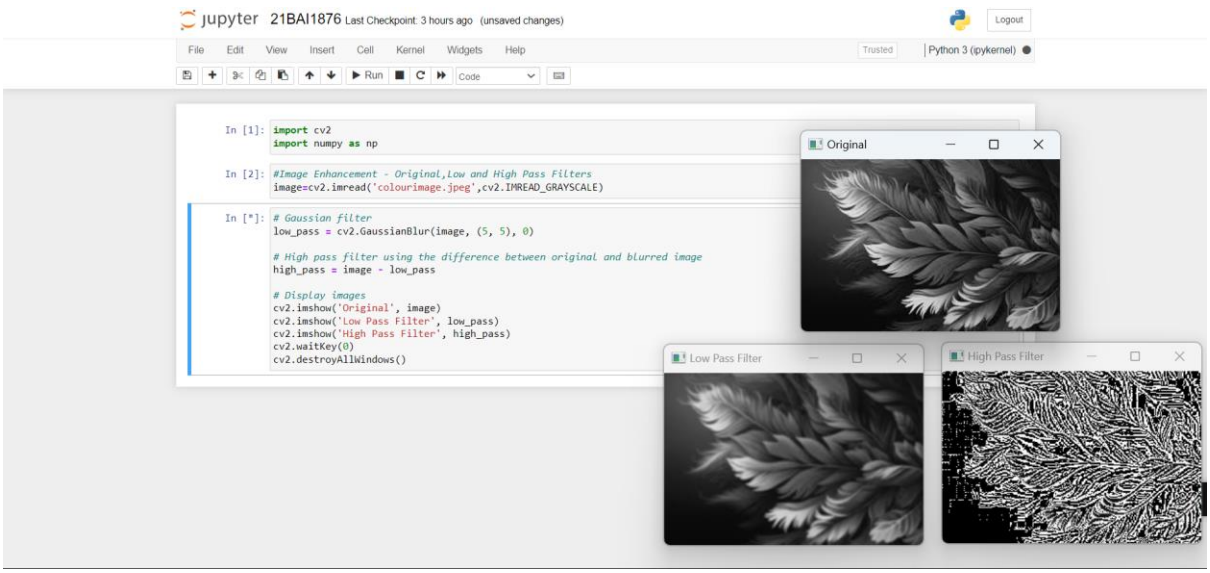
[Low Pass Filtering: Smooths the image, reduces high-frequency noise. High Pass Filtering: Enhances edges and fine details. Gaussian Filters are commonly used for both.]

CODE:

```
import cv2

import numpy as np

#Image Enhancement - Original,Low and High Pass Filters

image=cv2.imread('colourimage.jpeg',cv2.IMREAD_GRAYSCALE)

# Gaussian filter

low_pass = cv2.GaussianBlur(image, (5, 5), 0)



# High pass filter using the difference between original and blurred image

high_pass = image - low_pass



# Display images

cv2.imshow('Original', image)

cv2.imshow('Low Pass Filter', low_pass)

cv2.imshow('High Pass Filter', high_pass)

cv2.waitKey(0)
```

cv2.destroyAllWindows()

OUTPUT:



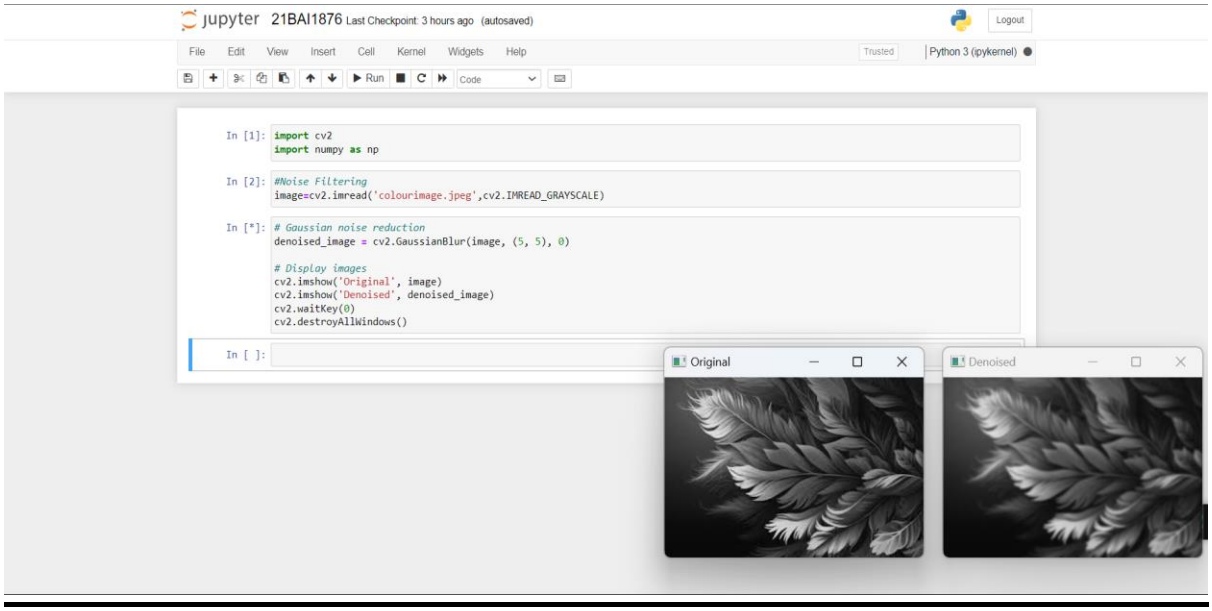
5. Noise Filtering – Gaussian Filters

[Gaussian filters are effective for reducing noise.]

CODE:

```
import cv2

import numpy as np

#Noise Filtering

image=cv2.imread('colourimage.jpeg',cv2.IMREAD_GRAYSCALE)

# Gaussian noise reduction

denoised_image = cv2.GaussianBlur(image, (5, 5), 0)

# Display images

cv2.imshow('Original', image)

cv2.imshow('Denoised', denoised_image)

cv2.waitKey(0)

cv2.destroyAllWindows()
```

OUTPUT:


jupyter 21BAI1876 Last Checkpoint: 3 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [1]:
import cv2
import numpy as np
In [2]:
#Noise Filtering
image=cv2.imread('colourimage.jpeg',cv2.IMREAD_GRAYSCALE)
In [*]:
# Gaussian noise reduction
denoised_image = cv2.GaussianBlur(image, (5, 5), 0)
# Display images
cv2.imshow('Original', image)
cv2.imshow('Denoised', denoised_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
In [ ]:
Original
Denoised