Department of Computer Science and Engineering (Data Science)
Image Processing and Computer Vision I (DJ19DSL603)

Lab 3: Image Enhancement in Spatial Domain using Point Processing Techniques

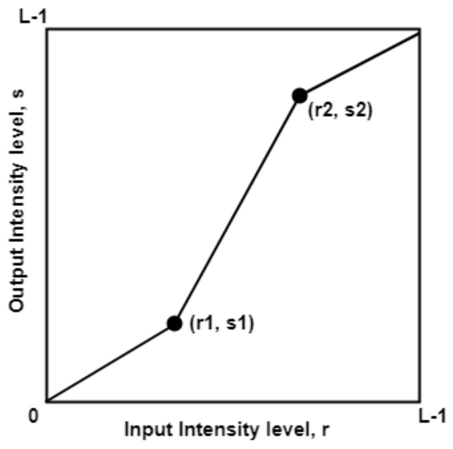
Aim: To perform image enhancement in spatial domain using point processing techniques: contrast stretching, log transformation, power law transformation

Theory:

1. **Contrast Stretching:**

Contrast stretching (often called normalization) is a simple image enhancement technique that attempts to improve the contrast in an image by 'stretching' the range of intensity values it contains to span a desired range of values, e.g. the the full range of pixel values that the image type concerned allows. It differs from the more sophisticated histogram equalization in that it can only apply a linear scaling function to the image pixel values. As a result the 'enhancement' is less harsh.

Below figure shows a typical transformation function used for Contrast Stretching.



By changing the location of points (r1, s1) and (r2, s2), we can control the shape of the transformation function. For example,

i. When r1 =s1 and r2=s2, transformation becomes a **Linear function**.
ii. When r1=r2, s1=0 and s2=L-1, transformation becomes a **thresholding function**.
iii. When (r1, s1) = (r_{min}, 0) and (r2, s2) = (r_{max}, L-1), this is known as **Min-Max Stretching.**
iv. When (r1, s1) = (r_{min} + c, 0) and (r2, s2) = (r_{max} – c, L-1), this is known as **Percentile Stretching**.

Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA : 3.18)

Department of Computer Science and Engineering (Data Science)
Image Processing and Computer Vision I (DJ19DSL603)
Lab 3: Image Enhancement in Spatial Domain using Point Processing Techniques

2. Log Transformation:

The log transformations can be defined by this formula:

$$s = c \log(r + 1)$$

Where s and r are the pixel values of the output and the input image and c is a constant. The value 1 is added to each of the pixel value of the input image because if there is a pixel intensity of 0 in the image, then log (0) is equal to infinity. So 1 is added, to make the minimum value at least 1.
During log transformation, the dark pixels in an image are expanded as compare to the higher pixel values. The higher pixel values are kind of compressed in log transformation.

3. Power Law Transformation:

The power law transformation, also called gamma transformation is a technique that uses a power-law function to adjust the pixel values of an image. It is versatile, as it allows the emphasis on certain intensity ranges or enhancing specific details in an image.

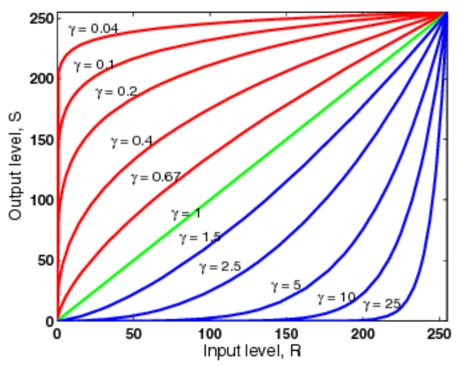
Formula of the gamma transformation equation is given as:

$$O = c*I^{\gamma}$$

In this context, the symbol c is used to represent the scaling factor, γ denotes the gamma correction value, and I and O respectively stand for the input and output pixel values.

If the value of gamma is greater than one, it stretches contrast in brighter areas and compresses the pixel values in the darker areas. If the value of gamma is smaller than one, it enhances contrast in dim areas and compresses the pixel values in the bright areas.The graphical representation of the function is given below:

Department of Computer Science and Engineering (Data Science)
Image Processing and Computer Vision I (DJ19DSL603)
Lab 3: Image Enhancement in Spatial Domain using Point Processing Techniques



Here, the x-axis and y-axis represent the input and output pixel values respectively and the shape of the graph varies with the gamma values.

Lab Assignments to complete in this session

Problem Statement: Develop a Python program utilizing the OpenCV library to enhance the images in spatial domain. The program should address the following tasks:

1. Read any low contrast image of your choice or generate one.
2. Display the before & after image(s) used in every task below.
3. Apply contrast stretching.
4. Apply Log transformation.
5. Apply Power law tranformation
6. Perform all the task for black and white images and on coloured RGB Images on all the three chanels.

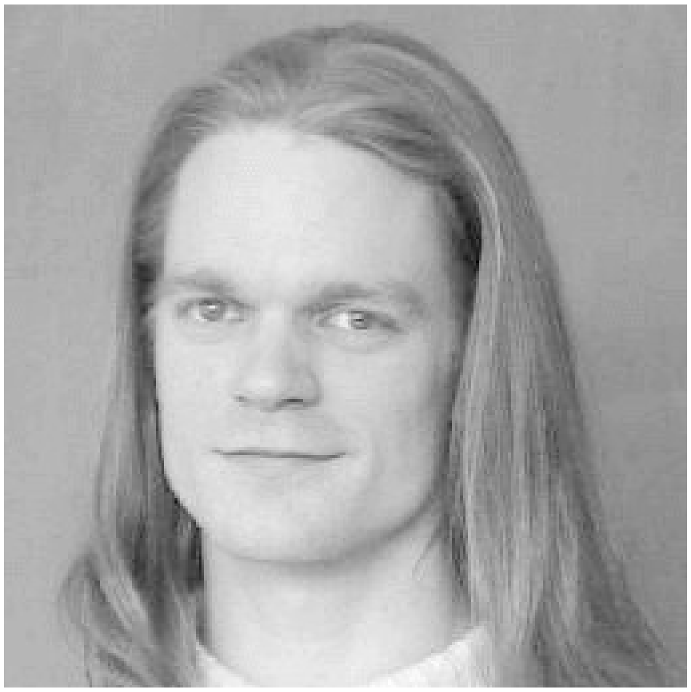
The solution to the operations performed must be produced by scratch coding without the use of built in OpenCV methods.

```
!pip install opencv-python
import cv2
```

```
Requirement already satisfied: opencv-python in
/usr/local/lib/python3.10/dist-packages (4.10.0.84)
Requirement already satisfied: numpy>=1.21.2 in
/usr/local/lib/python3.10/dist-packages (from opencv-python) (1.26.4)
```

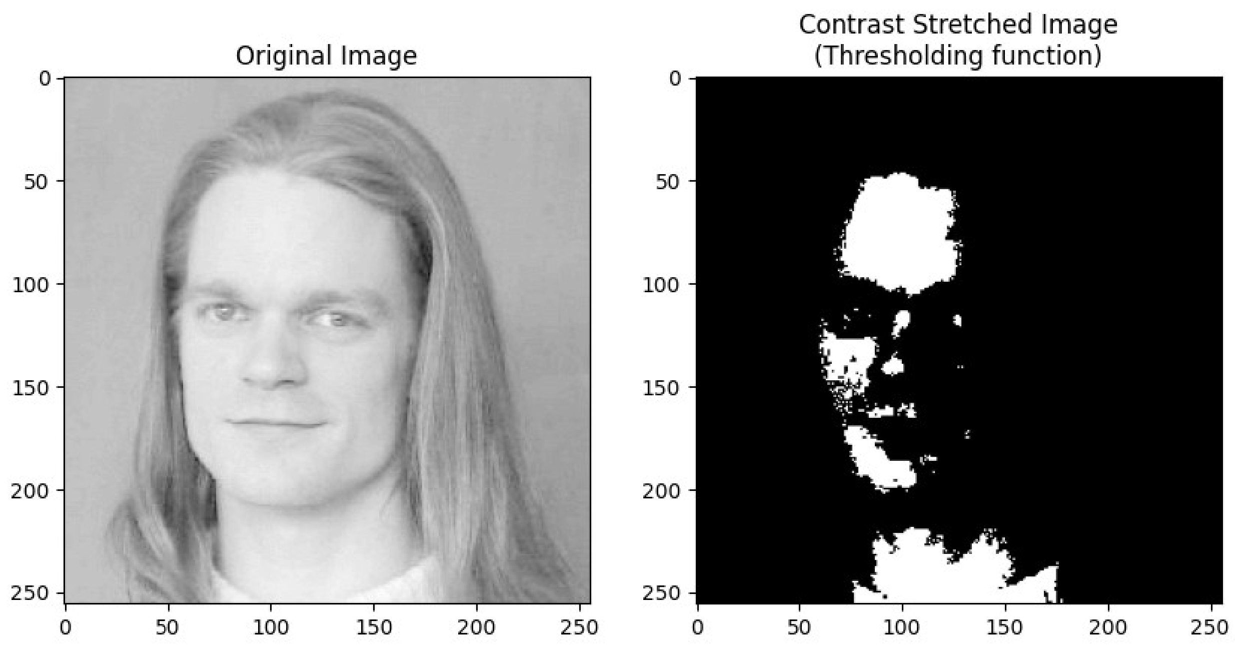
```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

def log_transform(image, c):


  image = image.astype(np.float32)


  transformed_image = c * np.log(image + 1)


  transformed_image = cv2.normalize(transformed_image, None, 255, 0,
cv2.NORM_MINMAX, cv2.CV_8U)
  return transformed_image

image = cv2.imread('/content/low-contrast-images.png')

color_image=cv2.imread('/content/color_image.webp')

log_transformed_image = log_transform(image, 0.2)


cv2_imshow(image)
cv2_imshow(log_transformed_image)
cv2.waitKey(0)
cv2.destroyAllWindows()

def power_law_transform(image, gamma, c):


  image = image.astype(np.float32)


  transformed_image = c * np.power(image, gamma)


  transformed_image = cv2.normalize(transformed_image, None, 255, 0,
cv2.NORM_MINMAX, cv2.CV_8U)
  return transformed_image
```

```
image = cv2.imread('/content/low-contrast-images.png')


power_law_transformed_image = power_law_transform(image, 2, 0.5)


cv2_imshow(image)
cv2_imshow( power_law_transformed_image)
```

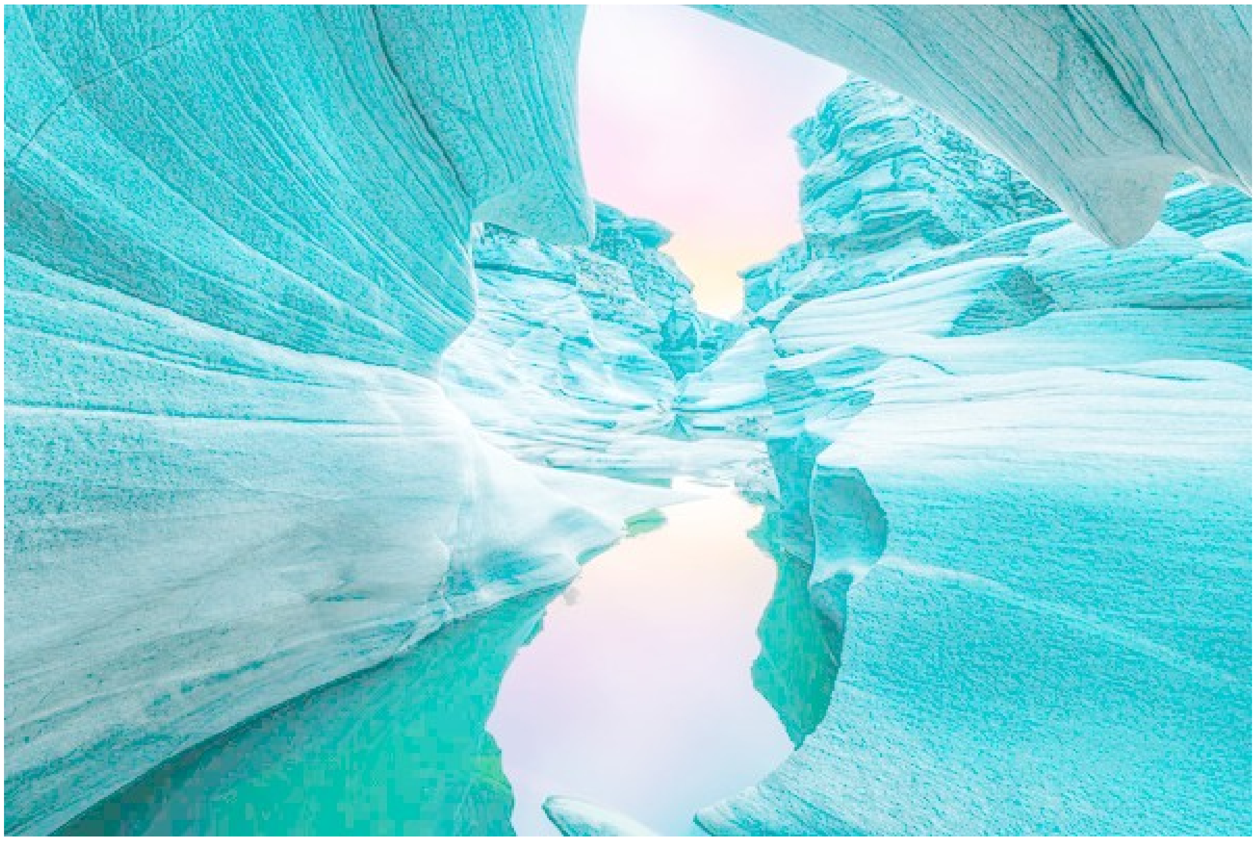


```python
import cv2
import numpy as np
import matplotlib.pyplot as plt


def contrast_stretching(image, r1, s1, r2, s2):
    output_image = np.zeros_like(image)


    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            pixel = image[i, j]
            if pixel < r1:
                output_image[i, j] = (s1 / r1) * pixel
            elif r1 <= pixel < r2:
                output_image[i, j] = ((s2 - s1) / (r2 - r1)) * (pixel
- r1) + s1
            else:
                output_image[i, j] = ((255 - s2) / (255 - r2)) *
(pixel - r2) + s2

    return output_image

def determine_stretching_type(r1, s1, r2, s2):

    if r1 == s1 and r2 == s2:
        return "Linear function"
    if r1 == r2 and s1 == 0 and s2 == 255:
        return "Thresholding function"


    r_min, r_max = np.min(image), np.max(image)
    if (r1, s1) == (r_min, 0) and (r2, s2) == (r_max, 255):
        return "Min-Max Stretching"


    c = 10
    if (r1, s1) == (r_min + c, 0) and (r2, s2) == (r_max - c, 255):
        return "Percentile Stretching"

    return "Custom Stretching"


image_path = "/content/low-contrast-images.png"
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)


r1, s1 = 240, 0
r2, s2 = 240, 255


stretched_image = contrast_stretching(image, r1, s1, r2, s2)


stretch_type = determine_stretching_type(r1, s1, r2, s2)
print(f"Stretching type: {stretch_type}")


plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(image, cmap='gray')
plt.title('Original Image')
plt.subplot(1, 2, 2)
plt.imshow(stretched_image, cmap='gray')
plt.title(f'Contrast Stretched Image\n({stretch_type})')
plt.show()
```

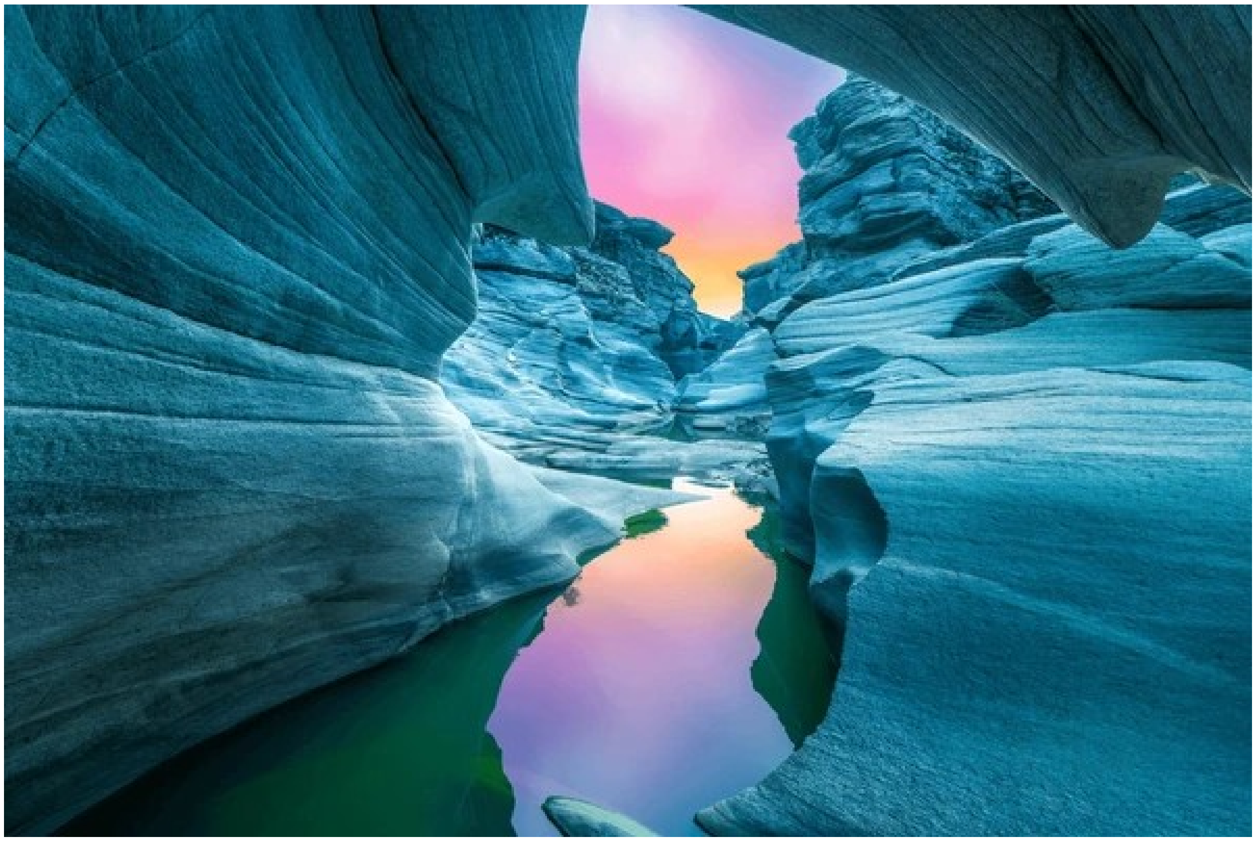
```
Stretching type: Thresholding function
```



```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow

def log_transform(image, c):

  image = image.astype(np.float32)

  # Use the 'image' argument instead of the global variable
'color_image'
  transformed_image = c * np.log(image + 1)

  transformed_image = cv2.normalize(transformed_image, None, 255, 0,
cv2.NORM_MINMAX, cv2.CV_8U)
  return transformed_image


color_image=cv2.imread('/content/color_image.webp')

log_transformed_image = log_transform(color_image, 0.2)


cv2_imshow(color_image)
cv2_imshow(log_transformed_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

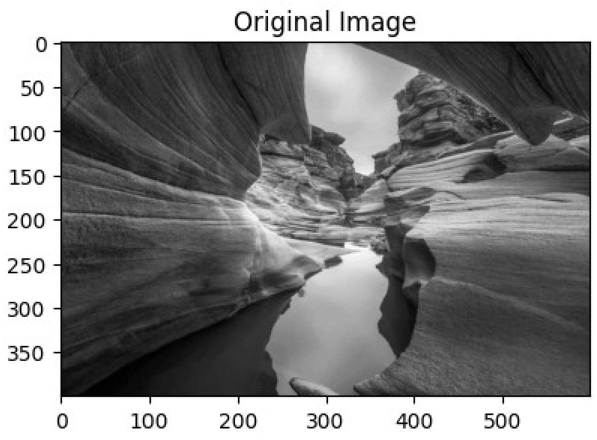


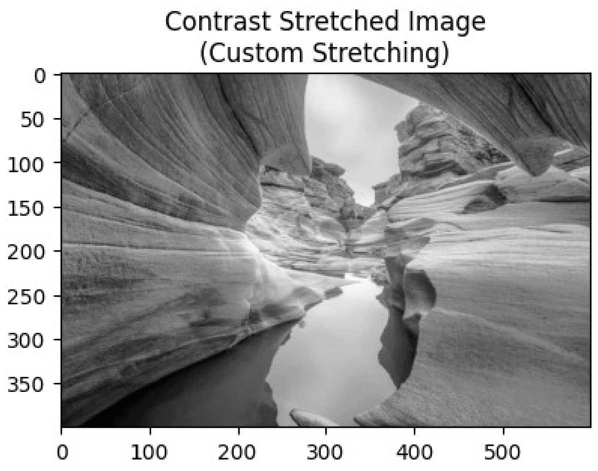
```
def power_law_transform(image, gamma, c):


  image = image.astype(np.float32)


  transformed_image = c * np.power(image, gamma)


  transformed_image = cv2.normalize(transformed_image, None, 255, 0,
cv2.NORM_MINMAX, cv2.CV_8U)
  return transformed_image


image = cv2.imread('/content/color_image.webp')


power_law_transformed_image = power_law_transform(image, 2, 0.5)


cv2_imshow(image)
cv2_imshow( power_law_transformed_image)
```



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


def contrast_stretching(image, r1, s1, r2, s2):
    output_image = np.zeros_like(image)


    for i in range(image.shape[0]):
        for j in range(image.shape[1]):
            pixel = image[i, j]
            if pixel < r1:
                output_image[i, j] = (s1 / r1) * pixel
            elif r1 <= pixel < r2:
                output_image[i, j] = ((s2 - s1) / (r2 - r1)) * (pixel
- r1) + s1
            else:
                output_image[i, j] = ((255 - s2) / (255 - r2)) *
(pixel - r2) + s2

    return output_image

def determine_stretching_type(r1, s1, r2, s2):
    if r1 == s1 and r2 == s2:
        return "Linear function"


    if r1 == r2 and s1 == 0 and s2 == 255:
        return "Thresholding function"


    r_min, r_max = np.min(image), np.max(image)
    if (r1, s1) == (r_min, 0) and (r2, s2) == (r_max, 255):
        return "Min-Max Stretching"


    c = 10
    if (r1, s1) == (r_min + c, 0) and (r2, s2) == (r_max - c, 255):
        return "Percentile Stretching"

    return "Custom Stretching"


image_path = "/content/color_image.webp"
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)


r1, s1 = 70, 120
r2, s2 = 40, 100


stretched_image = contrast_stretching(image, r1, s1, r2, s2)


stretch_type = determine_stretching_type(r1, s1, r2, s2)
print(f"Stretching type: {stretch_type}")


plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(image, cmap='gray')
plt.title('Original Image')
plt.subplot(1, 2, 2)
plt.imshow(stretched_image, cmap='gray')
plt.title(f'Contrast Stretched Image\n({stretch_type})')
plt.show()
```

```
Stretching type: Custom Stretching
```


Original Image
0
50
100
150
200
250
300
350
0
100
200
300
400
500


Contrast Stretched Image
(Custom Stretching)
0
50
100
150
200
250
300
350
0
100
200
300
400
500