Department of Computer Science and Engineering (Data Science)
Academic Year 2022-2023

Name: Sarvayga Singh

SAPID : 60009200030 **BATCH : K1**

AIM: To Perform Image Enhancement(SD) Point Processing Techniques: Contrast Stretching, Log Transformation, Power Law Transformation

THEORY:

1. Contrast Stretching

Low - contrast images can result from poor illumination, lack of dynamic range in the imaging sensor or wrong setting of a lens aperture during image acquisition. Contrast stretching is employed to increase the dynamic range of the gray levels in the image being processed.

The figure illustrates the form of the transformation function for contrast stretching:

The locations of points (r_1, s_1) and (r_2, s_2) control the shape of the transformation and thus the contrast of the output image.

If $r_1 = s_1$ and $r_2 = s_2$, the transformation is a linear function that produces no changes in the gray levels.

If $r_1 = r_2$, $s_1 = 0$ and $s_2 = L-1$, the transformation becomes a thresholding function that creates a binary image.

Intermediate values of (r_1, s_1) and (r_2, s_2) produce various degrees of spread in the gray levels of the output image, thus affecting its contrast.

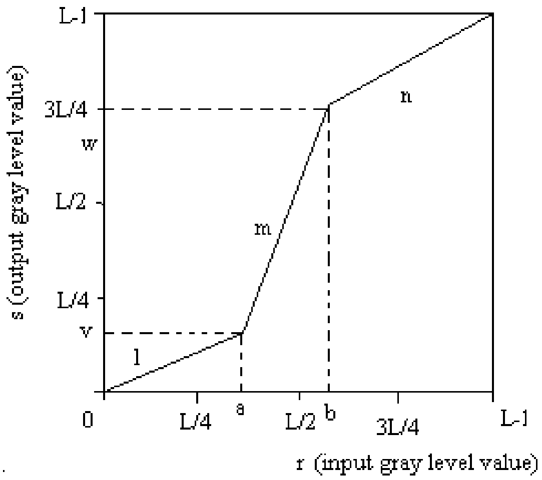


Fig 2. Contrast Stretching

In general, $r_1 \leq r_2$ and $s_1 \leq s_2$ is assumed to make the function single valued and monotonically increasing which preserves the order of gray levels thus preventing the creation of intensity artefacts in the processed image.

2. Dynamic Range Compression (Log Transformation)

The log transformations can be defined by this formula

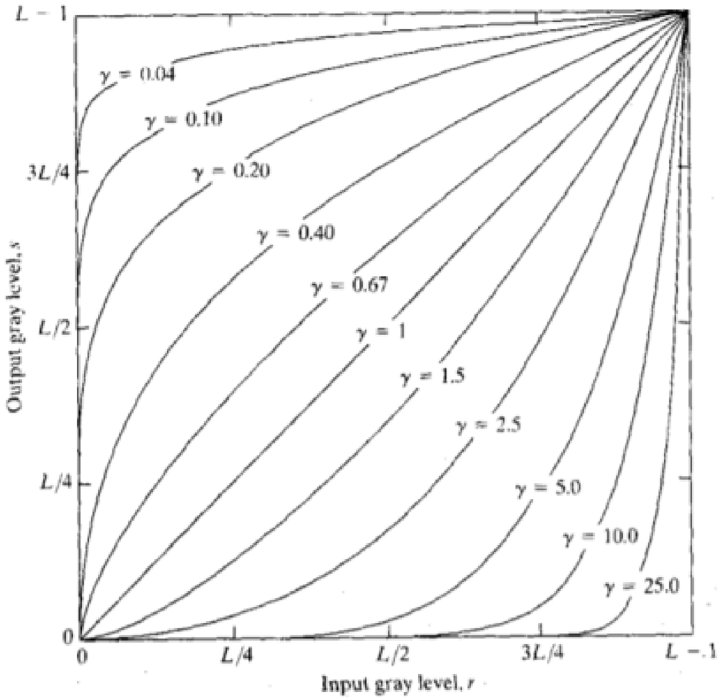
$$s = c \log(r + 1)$$

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:

$s = r^{\gamma}$



RESULT:

Sarvagya Singh
60009200030 - K1
IPCV -- lab2

In [1]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow
```

Contrast Stretching

In [ ]:
```
# Function to map each intensity level to output intensity level.
def pixelVal(pix, a, b, l, m, n):
  v = l*a
  w = m*(b-a) + v
  if (0 <= pix and pix <= a):
      return (v / a)*pix
  elif (a < pix and pix <= b):
      return ((w - v)/(b - a)) * (pix - a) + v
  else:
      return ((255 - w)/(255 - b)) * (pix - b) + w
```

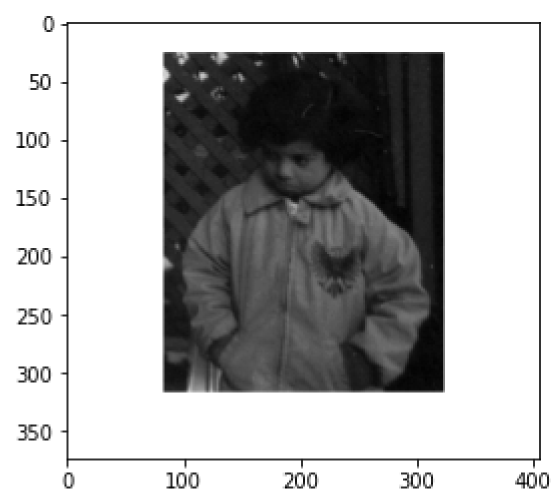
In the range

- l, n < 1
- m >= 1

In [ ]:
```
img = cv2.imread('/content/StandardDeviationBasedImageStretchingExample_01.png', 0)

a = 60
b = 140
l_1 = 0.1
m_1 = 1.5
n_1 = 0.6
```

In [ ]:
```
plt.imshow(img, cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7fc5403c50a0>
```



In [ ]:
```
img.min()
```

Out[ ]:
```
74
```

In [ ]:
```
img.max()
```

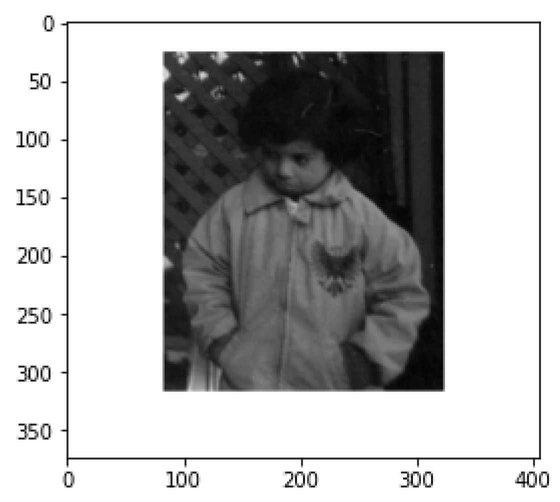
Out[ ]:
```
255
```

In [ ]:
```
pixelVal_vec = np.vectorize(pixelVal)
```

In [ ]:
```
contrast_stretched_1 = pixelVal_vec(img, a, b, l_1, m_1, n_1)
```

In [ ]:
```
plt.imshow(contrast_stretched_1, cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7fc541d04580>
```



In [ ]:
```
contrast_stretched_1.min()
```

Out[ ]:
```
27.0
```

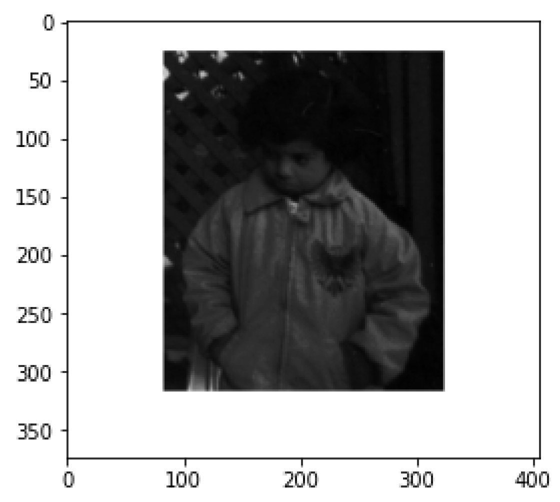
Outside the range

- l, n > 1
- m < 1

In [ ]:
```
a = 60
b = 140
l_2 = 1.5
m_2 = 0.5
n_2 = 2
```

In [ ]:
```
contrast_stretched_2 = pixelVal_vec(img, a, b, l_2, m_2, n_2)
plt.imshow(contrast_stretched_2, cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7fc5403465e0>
```



In [ ]:
```
contrast_stretched_2.min()
```

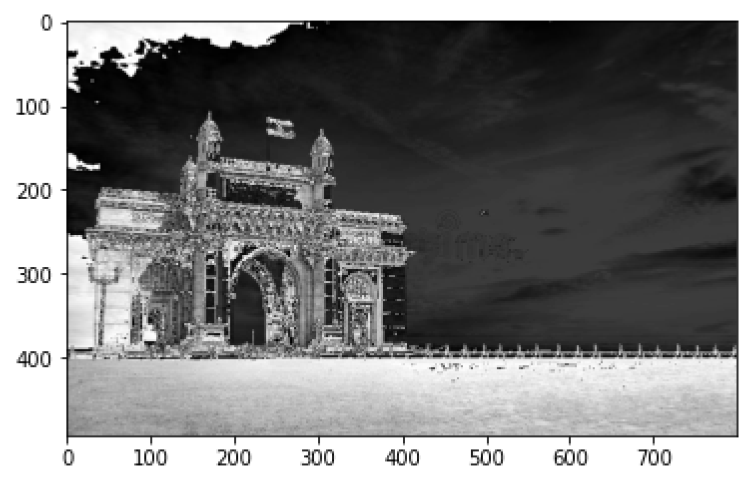
Out[ ]:
```
97.0
```

Log Transformation

In [ ]:
```
c = 150
log_transformed = c*np.log(img + 1)

log_transformed = np.array(log_transformed, dtype = np.uint8)

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

```
<ipython-input-25-012fe7bb06f9>:2: RuntimeWarning: divide by zero encountered in log
  log_transformed = c*np.log(img + 1)
```

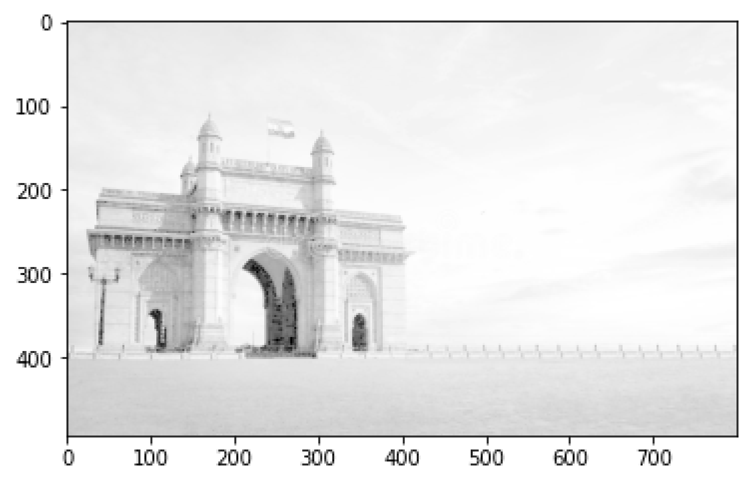
Out[ ]:
```
<matplotlib.image.AxesImage at 0x7fc542100610>
```



Power Law Transformation

In [ ]:
```
gamma_1 = 0.2
gamma_2 = 2.5

power_transformed_1 = img**gamma_1
plt.imshow(power_transformed_1, cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7fc5421b3820>
```



In [ ]:
```
power_transformed_2 = img**gamma_2
plt.imshow(power_transformed_2, cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7fc542192d30>
```

In [ ]:

Sarvagya Singh
60009200030 - K1
IPCV -- lab2

In [1]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow
```

Contrast Stretching

In [ ]:
```
# Function to map each intensity level to output intensity level.
def pixelVal(pix, a, b, l, m, n):
  v = l*a
  w = m*(b-a) + v
  if (0 <= pix and pix <= a):
      return (v / a)*pix
  elif (a < pix and pix <= b):
      return ((w - v)/(b - a)) * (pix - a) + v
  else:
      return ((255 - w)/(255 - b)) * (pix - b) + w
```

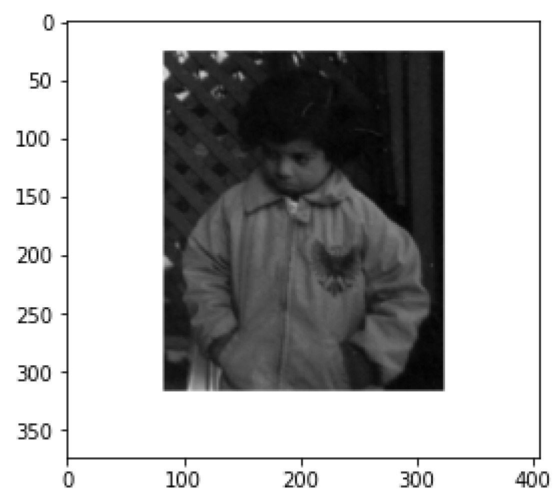
In the range

- l, n < 1
- m >= 1

In [ ]:
```
img = cv2.imread('/content/StandardDeviationBasedImageStretchingExample_01.png', 0)

a = 60
b = 140
l_1 = 0.1
m_1 = 1.5
n_1 = 0.6
```

In [ ]:
```
plt.imshow(img, cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7fc5403c50a0>
```



In [ ]:
```
img.min()
```

Out[ ]:
```
74
```

In [ ]:
```
img.max()
```

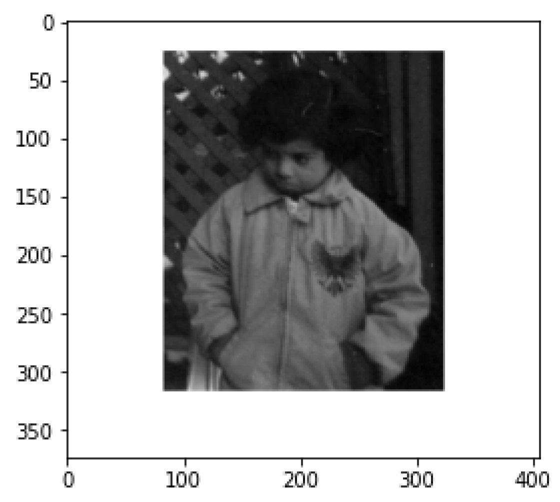
Out[ ]:
```
255
```

In [ ]:
```
pixelVal_vec = np.vectorize(pixelVal)
```

In [ ]:
```
contrast_stretched_1 = pixelVal_vec(img, a, b, l_1, m_1, n_1)
```

In [ ]:
```
plt.imshow(contrast_stretched_1, cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7fc541d04580>
```



In [ ]:
```
contrast_stretched_1.min()
```

Out[ ]:
```
27.0
```

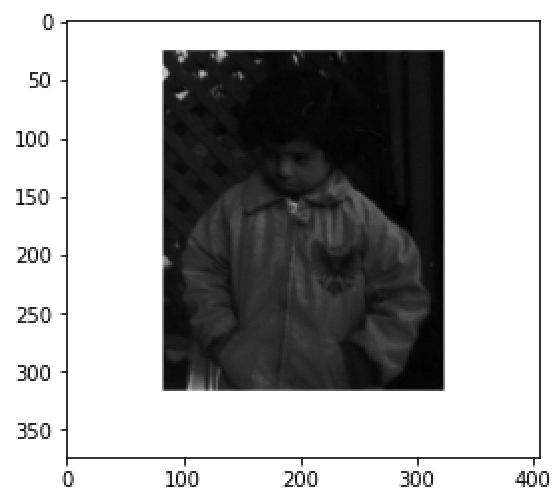
Outside the range

- l, n > 1
- m < 1

In [ ]:
```
a = 60
b = 140
l_2 = 1.5
m_2 = 0.5
n_2 = 2
```

In [ ]:
```
contrast_stretched_2 = pixelVal_vec(img, a, b, l_2, m_2, n_2)
plt.imshow(contrast_stretched_2, cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7fc5403465e0>
```



In [ ]:
```
contrast_stretched_2.min()
```

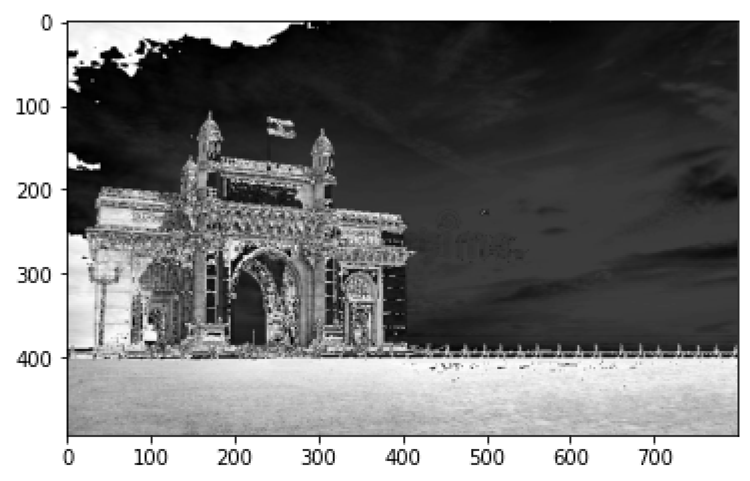
Out[ ]:
```
97.0
```

Log Transformation

In [ ]:
```
c = 150
log_transformed = c*np.log(img + 1)

log_transformed = np.array(log_transformed, dtype = np.uint8)

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

```
<ipython-input-25-012fe7bb06f9>:2: RuntimeWarning: divide by zero encountered in log
  log_transformed = c*np.log(img + 1)
```

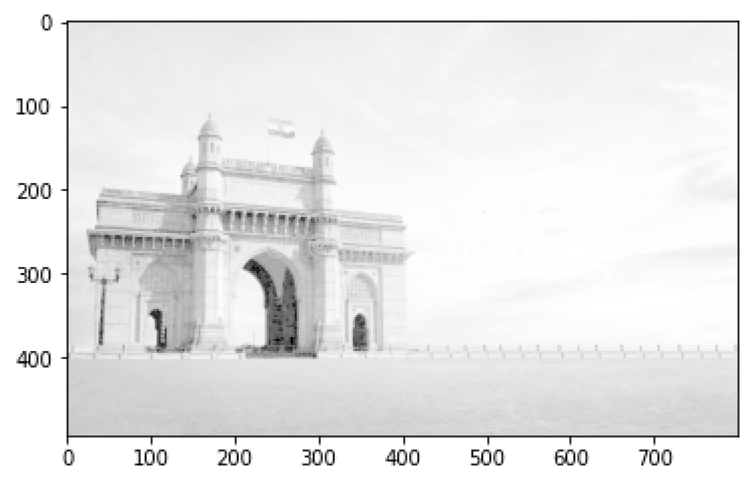
Out[ ]:
```
<matplotlib.image.AxesImage at 0x7fc542100610>
```



Power Law Transformation

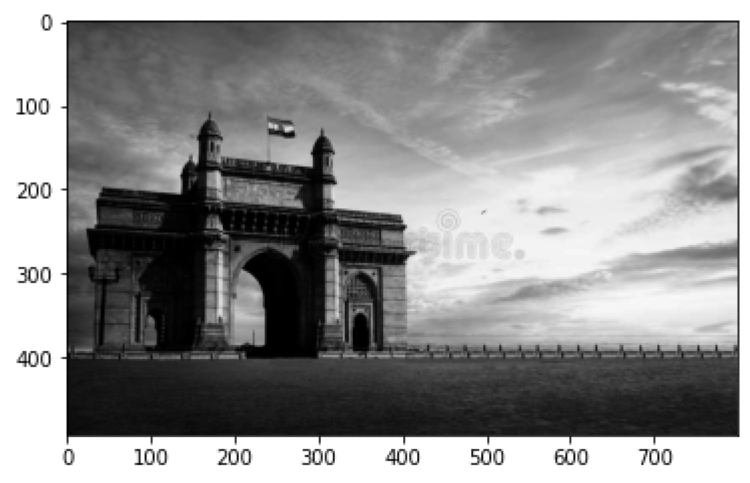
In [ ]:
```
gamma_1 = 0.2
gamma_2 = 2.5

power_transformed_1 = img**gamma_1
plt.imshow(power_transformed_1, cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7fc5421b3820>
```



In [ ]:
```
power_transformed_2 = img**gamma_2
plt.imshow(power_transformed_2, cmap='gray')
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x7fc542192d30>
```



In [ ]: